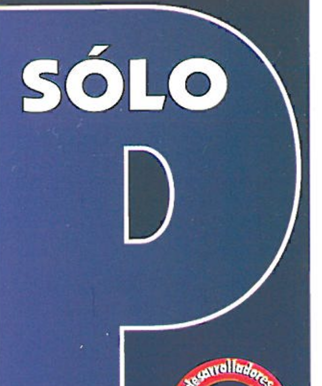
LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO



SÓLO PROGRAMADORES

Revista especializada para usuarios de PC

AÑO 3. Nº 27
1250 PTAS.
ARGENTINA 9'50 $
CHILE 3000 $
PORTUGAL 1500$

CD-ROM DE REGALO: SQL ANYWMERE Y VISUAL COMPONENTS

ANALIZAMOS LOS PROCESADORES DEL MERCADO

EXCEPCIONES EN DELPHI

WEB: ACTUALIZACIÓN DINÁMICA DE DOCUMENTOS

VISUAL J++

LA APUESTA DE MICROSOFT

Y además:

- Modo X de la VGA
- Sistemas operativos en red
- Control de flujo en Visual Basic

CURSOS PRÁCTICOS DE:

- Programación de demos
- Grandes sistemas
- Programación básica
- Visual C++ 4.0
- Redes Locales



8 413042 303299 00027



NOVIEMBRE 1996. Número 27
SÓLO PROGRAMADORES es una publicación de Tower Communications

Director Editor
Antonio M. Ferrer Abelló
aferrer@towercom.es
Redactor Jefe
Carlos Doral
Coordinador Técnico
Eduardo Toribio Pazos
Edición
Miguel Cabezuelo
Colaboradores
F. de la Villa, Fernando J. Echevarrieta, Pedro Antón. J. M. Martín, L. Martín, J. M. Peco, José C. Remiro, Daniel Navarro, Jorge del Río, Enrique De Alarcón, David Aparicio, M. Jesús Recio, Santiago Romero, Miguel Cubas, V. Cubas, C. Pérez

Jefe de Diseño
Fernando García Santamaría
Maquetación
Clara Francés
Tratamiento de Imagen
María Arce Giménez

Publicidad
Erika de la Riva (Madrid)
Tel.: (91) 661 42 11
Publicidad
Papín Gallardo (Barcelona)
Tel.: (93) 213 42 29

Suscripciones
Isabel Bojo
Tel. (91) 661 42 11 Fax: (91) 661 43 86
suscrip@towercom.es

Filmación
Filma Dos S.L.
Impresión
G. Reunidas
Distribución
SGEL

Director General
Antonio M. Ferrer Abelló
Director Financiero
Francisco García Díaz de Liaño
Director de Producción
Carlos Peropadre
Directora Comercial
Carmina Ferrer
Director de Distribución
Enrique Cabezas García
Asesor Editorial
Mario de Luis
Redacción, Publicidad y Administración
C/ Aragoneses, 7
28100 Pol. Ind. Alcobendas (MADRID)
Telf.: (91) 661 42 11 / Fax: (91) 661 43 86

Depósito legal: M-26827-1994
ISSN: 1134-4792

EDITORIAL



INTERNET Y LOS PROGRAMADORES

Internet se puede ver de diversas maneras. Muchas veces a nosotros, los programadores, nos preguntan en un bar a eso de las 3 de la mañana: “oye, tú que entiendes de ordenadores, ¿qué es eso de Internet?“, para inmediatamente continuar con un “¿los programadores tenéis algo que ver con ese asunto?” Bueno, lo complicado casi es responder a la primera cuestión. Posibles respuestas a esas horas de la mañana son :
Respuesta 1

- Internet lo inventaron los americanos porque los rusos iban a volar con sus bombas no sé que....
-Sí en cuanto a programación... , y sueltas un rollo de aplicaciones cliente servidor, lo mezclas con algo de Browsers y applets Java y protocolos TCP/IP...
(ya te has quedado solo en el bar)

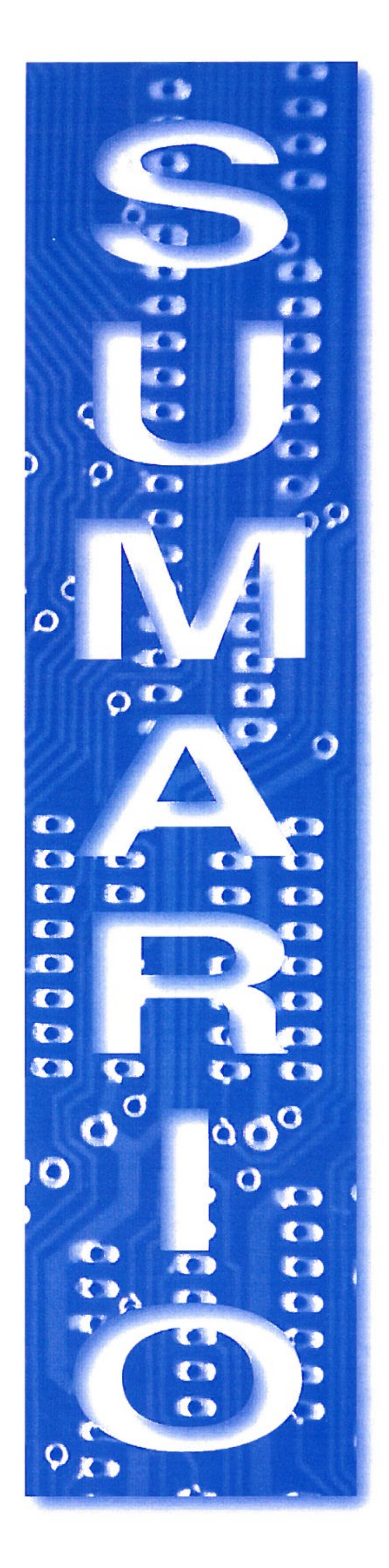
Respuesta 2
-Internet es la red de redes, te comunicas a distancia con otros usuarios y hace que tendamos todos hacia la aldea global.
- No, los programadores no tenemos mucho que decir en este asunto.
(la gente lo entiende, y alguien hábilmente vuelve a sacar el tema del fútbol)

Respuesta 3
-Internet está bien para ver la página de Playboy y otras cosillas.
-Es muy fácil, pones www.playboy.com y ya está.
(respuesta que quieren los amigotes oír. Alguno sugiere la posibilidad de ir a tu casa y asistir a una demostración didáctica in situ, asombrosamente repite una y otra vez “w playboy punto cong”)

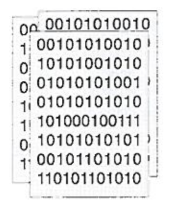
Respuesta 4
-Internet es hoy un gran tablón de anuncios, y mañana una herramienta de trabajo para casi todos, desde vendedores, repartidores, secretarias, hasta el más importante ejecutivo, muchos de vosotros lo utilizareis antes de lo que creéis.
-No está claro el tema de quién se va a imponer en este asunto, todo el mundo habla de Java, pero desde luego hoy por hoy no es la herramienta perfecta, las grandes empresas en estos momentos luchan por imponer sus componentes y sus estándares.
(Es para mí una buena respuesta, pero los ajenos a la materia no entenderán nada)

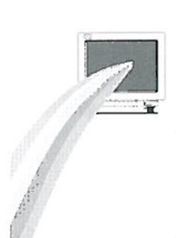
En resumen, ¿qué es Internet ? es la pregunta del millón, y cada vez que nos la hacen nos ponen en un aprieto, pues la gente desea una respuesta breve y concisa, lo cual en este caso se me antoja imposible. Pero en la redacción de Sólo Programadores, lo que nos preguntamos es ¿cómo va a afectar a los programadores Internet? o más concretamente ¿cuántos vamos a trabajar o desarrollar nuestra actividad laboral gracias a la red de redes? Lo que sí que está claro, y que en nuestra profesión nadie duda, es aquello de renovarse o morir. Habrá que estar al día en Java, HTML, y mirarse las distintas aplicaciones comerciales que vayan apareciendo en el mercado, no sea que de repente una se convierta en la estándar y nos cierre alguna puerta interesante.

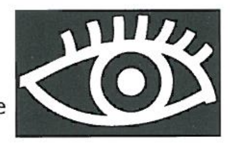
De cualquier manera es un momento muy interesante para la red, ¿impondrá Microsoft su Explorer?, ¿lo hará Netscape?, ¿es Java el futuro, o de momento es muy limitado, sobre todo gráficamente? A nosotros lo que nos importa en mayor medida son las respuestas a las siguientes cuestiones: ¿Cuántos programadores podrán desarrollar su profesión gracias a Internet? Permaneceremos expectantes.


SUMARIO

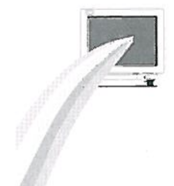

27

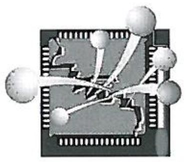


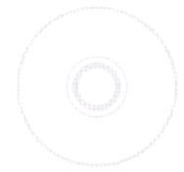


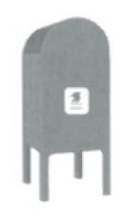










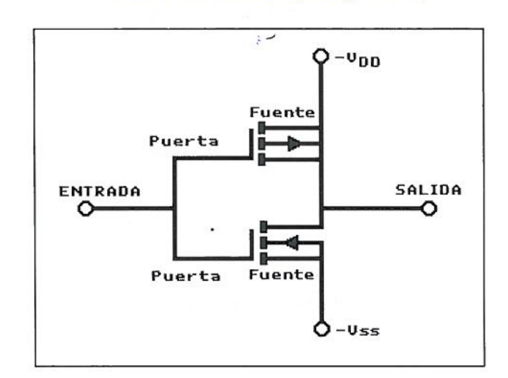


VISUAL J++ (Pág 66)



Las grandes compañías apuestan por Internet, la tarta es muy sabrosa y todos quieren su pedazo, en el interior el análisis de Sólo Programadores de la versión beta de Visual J++. ¿Es Java el futuro, o lo es Visual J de Microsoft ? Pronto saldremos de dudas.

PC INTERNO (Pág. 44)



Que los procesadores han experimentado una evolución vertiginosa en los últimos 20 años no escapa a nadie, en este número 27 dedicamos unas páginas a realizar un repaso a la historia más reciente de los procesadores. También queremos destacar este mes el curso de programación, esta vez un estudio sencillo pero muy didáctico sobre el teclado.



OBJECTSUITE
Librerías gráficas en C++

ESRI ESPAÑA GEOSISTEMAS ha comenzado a distribuir ObjectSuite, un software de VISUAL NUMERICS, en nuestro país. El paquete, disponible para plataformas Windows NT/95, Windows con Win32s y Unix, consiste en una serie de librerías numéricas y gráficas en C++ orientadas a objetos que permiten un rápido desarrollo de aplicaciones. Este paquete aporta funciones matemáticas avanzadas, estadística básica, procesamiento de señales y gráficos financieros.

ObjectSuite consta de tres extensiones: un módulo de matemáticas para C++, que es un conjunto de librerías numéricas para desarrollos en C++ que precisen operaciones matemáticas avanzadas y estadística básica; un módulo de procesamiento de señales para C++ que utiliza la potencia y flexibilidad del lenguaje C++ para reducir enormemente el tiempo de programación, así como la cantidad de código necesario y un módulo de gráficos financieros para C++ que aporta un interfaz orientado a objetos para visualizar gráficos a cualquier nivel y permite visualizar gráficas de secciones en dos y tres dimensiones. Cada uno de los tres módulos ofrece una amplia documentación *on-line*.

Para más información:
ESRI ESPAÑA GEOSISTEMAS
Tel: (91) 559 43 75
Fax: (91) 559 70 71

JAVA BEANS
Componentes API para Java

Recientemente ha sido presentado el proyecto Java Beans para definir una serie de componentes API estándar para la plataforma Java. El objetivo es facilitar el desarrollo de componentes software reutilizables que el usuario final podrá enlazar mediante las herramientas de desarrollo de la aplicación.

Los nuevos Java Beans API serán portables entre todas las plataformas que soporten Java, y podrán conectarse mediante los componentes existentes como Microsoft ActiveX, OpenDoc y Netscape LiveConnect. Los componentes software podrán ser usados en una gran variedad de entornos, como Internet Explorer, Netscape Navigator, HotJava, CyberDog, Visual Basic, Word, Claris Works... y trabajarán con una amplia gama de herramientas de creación de aplicaciones. Entre las novedades que aportará la iniciativa Java Beans se incluye el soporte de ficheros JAR y la llamada a métodos remotos.

Con un entorno compacto y potente, la plataforma Java proporciona a los programadores la potencia de un lenguaje orientado a objetos mientras la complejidad de crear objetos y registrar procesos requerida por otros lenguajes y entornos de programación. Esta potencia será aprovechada por Java Beans para facilitar enormemente la creación de aplicaciones y componentes.

Para más información:
http://java.sun.com

JAVA CLASS FOR LOGIC SERVER
Nueva clase para Java de Azmi Inc.

Amzi! Inc. ha anunciado recientemente una nueva clase Java para su servidor Amzi! Logic Server. Esta herramienta permite incrustar agentes y componentes inteligentes en aplicaciones Java, clientes y servidores. Estos componentes pueden diagnosticar problemas, analizar documentos Web, indicar configuraciones recomendadas, informar sobre la actividad del monitor y muchas otras funcionalidades. Esta nueva clase está implementada como una librería DLL que corre bajo Windows NT con el entorno de desarrollo Java de Sun. La clase proporciona alrededor de 50 métodos para filtrar y administrar reglas y factores en bases lógicas. El servidor Logic Server forma parte de la implementación Amzi! Prolog + Logic Server, que ha sido designado para incrustar componentes y agentes inteligentes, así como bases lógicas, en una gran variedad de aplicaciones cliente/servidor. Al mismo tiempo, Amzi! ha anunciado también la disponibilidad inmediata de Amzi! Logic Explorer, que permitirá al usuario aprender la tecnología de bases lógicas y desarrollarlas. Esta herramienta ofrecerá un entorno de desarrollo integrado para Windows con un editor para escribir los factores y las reglas, un intérprete para filtrar y administrar la base lógica y un depurador para monitorizar la ejecución de los filtros.

Para más información:
Amzi! Inc.
http://www.amzi.com/

JDK 1.1
Nuevas mejoras para JAVA

Recientemente ha sido anunciada la nueva versión la herramienta de programación Java Development Kit (JDK). El nuevo compilador incluye nuevas implementaciones que mejorarán la funcionalidad y la calidad de Java. Entre las nuevas mejoras se encuentra el soporte de ficheros JAR, que permite guardar los applets Java y todos sus componentes en un único fichero, la posibilidad de incorporar distintos niveles de seguridad en las aplicaciones, la llamada a métodos remotos y nuevas clases soportadas.

Asimismo, incluirá el *Java Database Connectivity* (JDK), un interfaz estándar de acceso a bases de datos que ofrecerá un acceso uniforme a una gran cantidad de bases de datos relacionales. Dentro de este mismo interfaz, JDK proporcionará una base común indicando las herramientas de alto nivel y los entornos que puedan ser desarrollados.

Para más información:
http://java.sun.com

INTERBASE 4.2
Nueva versión del servidor de bases de datos de Borland

Borland International Inc. ha anunciado recientemente la versión 4.2 de InterBase, su servidor escalable de Bases de datos SQL para plataformas Windows 95 y Windows NT. InterBase 4.2 proporciona mejoras en la ejecución y uso de recursos superior, en comparación con las versiones previas, así como la implementación de una versión mejorada de la arquitectura InterBase SuperServer. El paquete incluye también los drivers ODBC 2.5, para conectividad de bases de datos, para Windows 95 y Windows NT y librerías cliente de 32 bits.

InterBase 4.2 continúa proporcionando plenas compatibilidades de servidor a través de entornos Windows y Unix y mantiene los anteriores formatos de bases de datos, lenguaje SQL y API. Estas compatibilidades entre plataformas permite a los desarrolladores escribir sus aplicaciones una vez, y desplegarlas más tarde en sistemas Windows o Unix.

Las nuevas funcionalidades de InterBase 4.2 incluyen, además, el desarrollo para grandes sistemas multiusuario, herramientas SQL de 32 bits, los antes mencionados drivers ODBC de 32 bits, código idéntico para Windows 95 y Windows NT, optimización para el nuevo Windows NT 4.0 y soporte de drivers ODBC de Java para InterBase e InterClient.

Para más información:
http://www.borland.com

SFTTREE/OCX
Controles ActiveX para entornos visuales de programación

Softel Vdm ha lanzado recientemente SftTree/OCX 2.0, un control ActiveX que ofrece posibilidades de visualización de datos por jerarquías como múltiples columnas, selección múltiple, múltiples líneas por ítem, cabeceras de columnas (con inclusión opcional de botones o bitmaps), columnas modificables por el usuario, visualización de ítems en 3D, cálculo del área de scroll horizontal, etc...

Se pueden usar varios bitmaps para cada ítem, incluyendo imágenes de botón ó de celdas para que puedan ser expandidas o contraídas según hayan sido definidas por el usuario. También ofrece un soporte drag-and-drop para facilitar las tareas realizadas mediante el método de arrastrar y soltar, incluyendo el scroll vertical durante este proceso. Al mismo tiempo, SftTree/OCX incluye un extenso manual de referencia, además de una amplia ayuda online, que hará mucho más fácil su uso.

SftTree/OCX incluye controles ActiveX de 16 y 32 bits, con soporte para Windows 3.1/95/NT. Los controles pueden ser usados en Visual Basic, Delphi, Visual C++, Ms Access y el resto de herramientas que soporten las especificaciones ActiveX.

La versión para desarrolladores de SftTree/OCX requiere alrededor de 7 Mb de espacio en disco y 400 Kb para los controles ActiveX (incluyendo las librerías MFC y los controles OLE). Por supuesto, estos controles pueden ser distribuidos sin pago de royalties con las aplicaciones desarrolladas.

Para más información:
http://www.softelvdm.com

VISUAL BASIC APPFRAMEWORK
Herramienta para Visual Basic de Crescent Software

Crescent Software ha anunciado Visual Basic AppFramework, una herramienta para Visual Basic que reside directamente dentro del propio entorno y consolida los pasos del proceso de programación. Al usar VB AppFramework el usuario dispondrá de un completo entorno de aplicación personalizado para comenzar los nuevos proyectos, añadir colecciones de controles a cualquier formulario, personalizar formularios y controles sin acceder a las propiedades de ventana, insertar los formularios más utilizados en proyectos existentes y "limpiar" los diseños descartados.

VB AppFramework incluye una librería de diseños profesionales de formularios y funciones libres de royalties. Con esto, el usuario podrá utilizar los diseños incluidos, modificarlos y personalizarlos según sus necesidades, o reconstruir los objetos existentes incluyendo nuevos estilos.

Al mismo tiempo, VB Appframework reduce el tiempo de desarrollo, por lo que los programadores podrán ahora realizar en pocos segundos lo que antes llevaría días u horas. La razón de este ahorro de tiempo es la posibilidad de construir librerías de estilos reutilizables; automatización de la captura de patrones y estilos de los objetos existentes; la organización automática de los patrones, referencias OLE y ficheros externos y la clonación instantánea de formularios, controles y estilos. VB AppFramework está disponible a un precio aproximado de 32.000 pesetas, y puede adquirirse a través de los distribuidores oficiales de Crescent o mediante correo electrónico a la dirección *crescent@progress.com*.

Para más información:

http://www.progress.com/crescent

SYSTOOLS 1.0
Librería de funciones de bajo nivel para Delphi

TurboPower Software ha lanzado SysTools 1.0, una librería orientada a sistemas de bajo nivel para el entorno de desarrollo Delphi 1.0/2.0. SysTools proporciona rutinas de indexado para el manejo de cualquier tipo de datos, funciones para manipular cadenas y fechas, aritmética binaria natural de alta precisión, rutinas para la modificación de ficheros y registros INI, y muchos otros tipos de funciones. SysTools También soporta compatibilidad 16-32 bits.

Al contrario que los controles ActiveX, OCX, VBX o Dlls, SysTools compila directamente en la aplicación.

Para más información:

http://www.tpower.com

S-DESIGNOR 5.1
Powersoft anuncia la nueva versión de sus herramientas de diseño y análisis

Powersoft, la división de herramientas de desarrollo de Sybase, ha anunciado una nueva versión de su conjunto de herramientas de diseño y análisis cliente/servidor S-Designor. La nueva revisión (5.1) incluye tecnología de generación de objetos Visual Basic, así como la versión Desktop de AppModeler, que incorpora diseño de bases de datos y tecnología de generación de objetos para las principales herramientas de desarrollo cliente/servidor y bases de datos del mercado. La línea S-Designor facilita la creación eficiente de aplicaciones y bases de datos de alta calidad para entornos cliente/servidor, y permite a los usuarios el análisis de los procesos de negocio.

Además de la versión Desktop de AppModeler, S-Designor 5.1 incluye tres módulos que permiten a los equipos de proyecto adaptar un conjunto de herramientas de modelado de aplicaciones para satisfacer sus necesidades de desarrollo de forma económica.

Los módulos de los que se compone la nueva versión de S-Designor son: DataArchitect, para diseño y construcción de bases de datos a niveles físico y conceptual; ProcessAnalyst, para descubrimiento de datos; AppModeler, para modelado físico de bases de datos y generación de objetos y aplicaciones y MetaWorks, para capacidades de diccionario multiusuario.

La inclusión de tecnología de generación de objetos Visual Basic extiende la estrategia de generación de objetos de Powersoft, y permite a los desarrolladores la generación de objetos personalizables. Esta nueva tecnología acelera los desarrollos en Visual Basic, a la vez que mejora la calidad de las aplicaciones cliente/servidor de forma global, ya que automatiza muchas de las tareas comunes de programación asociadas a desarrollos de este tipo.

Para más información:

http://www.sybase.com
http://www.powersoft.com

TOWEREIFFEL 2.0
Entorno de desarrollo orientado a objetos

Tower Technologies ha anunciado TowerEiffel Release 2.0, un entorno de desarrollo orientado a objetos para construir aplicaciones, controles y sistemas reutilizables. Esta herramienta está destinada a desarrolladores de software que realicen sistemas cliente/servidor con complejos requerimientos. TowerEiffel 2.0 proporciona soporte completo para el lenguaje de programación Eiffel, e incluye un entorno integrado de desarrollo y sus correspondientes herramientas, tecnología de compilación y librerías reutilizables para interfaces gráficos, estructuras de datos y soporte de proceso distribuido.

TowerEiffel 2.0 incluye soporte para interoperabilidad directa con C, C++ y Objective-C, serialización para la distribución de procesos, soporte de librerías estáticas precompiladas y localización de ficheros de objetos. TowerEiffel Proffesional estará disponible para plataformas Windows NT/95 y Linux.

Para más información:

http://www.twr.com

MACROMEDIA EXTREME 3D 2
Herramienta de diseño para Internet

Macromedia ha lanzado recientemente su nueva herramienta de diseño, multimedia e Internet para 3 dimensiones. Esta es una herramienta pensada para desarrolladores de multimedia y profesionales de vídeo que quieran incluir las 3D en sus páginas Shockwave, diseños profesionales o producciones multimedia. Entre sus nuevas características destaca el soporte Internet para formatos VRML, la compatibilidad con Director, Authorware y Shockwave y la interactividad con Direct 3D de Microsoft. Esta nueva herramienta está disponible en inglés al precio de 63.000 pesetas para Windows 95, Windows NT, Macintosh 68K y Power PC.

VIVID RESOLUTION
Generador de aplicaciones para Visual Basic y HTML

Intek Technologies ha lanzado recientemente Vivid Resolution for Visual Basic and HTML, un generador de aplicaciones que genera automáticamente programas totalmente funcionales (incluyendo sus proyectos, formularios, objetos de acceso a datos y código) para aplicaciones visuales y modelos de datos para Visual Basic Enterprise Edition 4.0 y HTML. El generador HTML marca el modelo de datos con el código apropiado para que pueda ser leído por cualquier navegador o aplicación HTML estándar. Los datos marcados pueden ser mezclados con alguno de los formatos disponibles vía Microsoft Internet Database Connector (IDBC).

Para más información:

http://www.intekinc.com

LENGUAJE F
Nueva revisión de Fortran

Imagine1, junto con el Numerical Algorithms Group (ANG), Fujitsu y Absoft, ha lanzado el lenguaje de programación F. Este nuevo lenguaje es un subconjunto de la más reciente versión de Fortran. F conserva las anteriores funcionalidades de módem de Fortran (por ejemplo, los módulos de abstracción de datos) y añade otras, como equivalencia, que es más difícil de enseñar, usar o depurar. El resultado es un lenguaje seguro y portable. F puede ser usado por los usuarios de Fortran 77 como transición hacia los nuevos conceptos de Fortran 95 o High Performance Fortran (HPF).

F está disponible para las plataformas UNIX y Linux, Macintosh 68K o Power PC y entornos PC bajo Windows 95/NT. Para el nuevo lenguaje, NAG ha proporcionando la tecnología de compilación usada en las implementaciones de UNIX y Linux, Fujitsu la tecnología de compilación usada en las implementaciones de Windows y Absoft la tecnología de compilación usada en las implementaciones de Macintosh.

Se puede obtener una versión para uso educativo de F para Linux vía FTP a través de *ftp.swcp.com* en el directorio *pub/walt/F* del Web de la compañía.

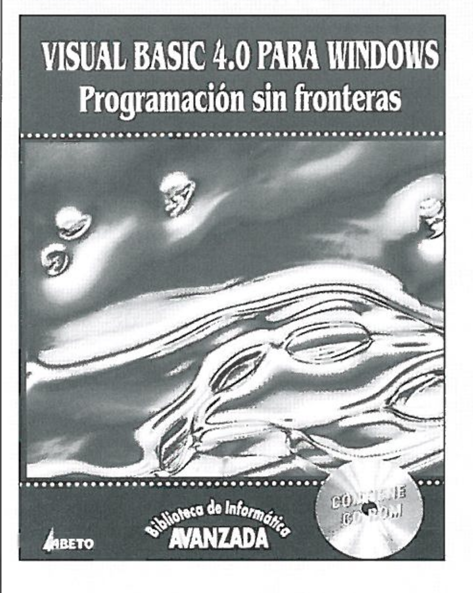
Para más información:
http://www.imagine1.com/imagine1/

NOVEDADES

POWERSOFT ANUNCIA NETIMPACT STUDIO

NetImpact Studio es un completo entorno para el desarrollo de aplicaciones profesionales de negocio para la Web. Permite el desarrollo de aplicaciones Web dinámicas, basadas en datos utilizando HTML, JavaScript, componentes (como ActiveX o Plug-ins) y sentencias SQL o procedimientos de aplicaciones de servidor. Este completo entorno de desarrollo incluye todas las herramientas y servicios necesarios para desarrollar, testear y desplegar aplicaciones de negocio para la Web y para Intranets corporativas NetImpact Studio está planificado que entre en fase Beta este último trimestre de año. Como principales características podemos subrayar : Completo entorno de desarrollo: Diseñado específicamente para el desarrollo de aplicaciones de negocio, ofrece un soporte abierto y extensible a los principales browsers, servidores web, aplicaciones de servidor, lenguajes y bases de datos del mercado. HTML fácil : NetImpact Studio ofrece a los desarrolladores un editor/browser HTML con capacidad WSYWYG, con soporte HTML 3.2 y características avanzadas como hojas estilo cascada, tablas y macros. Data Smart: NetImpact Dynamo permite a los desarrolladores una sencilla y rápida gestión de Web sites con contenidos dinámicos . Server Savvy : Es una herramienta de desarrollo Web que permite acceso abierto a múltiples servidores de aplicaciones. Rico en componentes : Ayuda a los desarrolladores a aprovechar por completo componentes de terceros fabricantes, entre los que se incluyen Java applets, Plug-ins de Netscape o los controles de ActiveX de Microsoft.

LIBROS



Visual Basic 4.0 para Windows: Programación sin fronteras

El objetivo de este libro es ayudar con los primeros pasos en Visual Basic 4.0 y dar a conocer las nuevas potencialidades de este lenguaje. Dado que con los lenguajes de programación visual, la potencia y versatilidad de las aplicaciones que pueden diseñarse se ha visto notablemete incrementada. En este sentido, Visual Basic se ha colocado al frente del mercado, por prestaciones y comodidad de uso.

En este libro se encuentra información sobre temas como: Utilización de Visual Basic, conceptos generales, instalación etc. El entorno de la herramienta : la barra de menús, la ventana de propiedades, y en general todo lo que se necesita para crear una aplicación.

Los controles del lenguaje : desde los más simples a controles OLE y MCI.

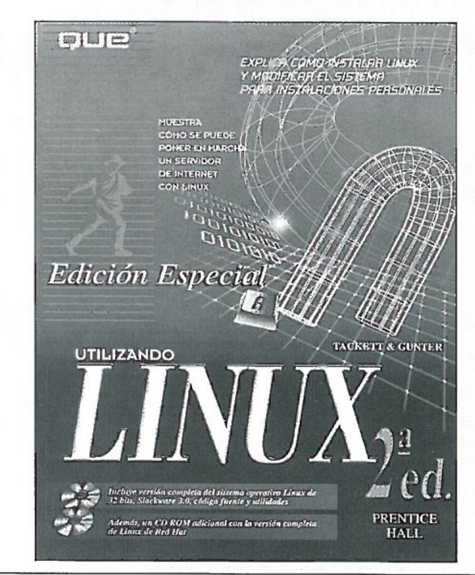
La gestión de sus proyectos : creación de una aplicación y trabajo con formularios.

Herramientas de depuración y control de errores que Visual Basic proporciona.

Editorial: Abeto
Autores: María Calvo y María Cuadra
193 páginas y CD-ROM
Idioma: Castellano

Utilizando Linux, edición especial

Utilizando Linux Segunda edición, o edición especial de Prentice Hall, es un perfecto manual para todos aquellos que quieran introducirse en el mundo del sistema operativo Linux. El presente libro constituye una perfecta ayuda para instalar, configurar y trabajar con este potente sistema operativo, de moda en los últimos tiempos. En el interior de sus páginas, se encuentra toda la información referente a desde cómo ejecutar aplicaciones Linux, realizar copias de seguridad, hasta cómo configurar una red TCP/IP y utilizar el correo electrónico, también hallará toda la información necesaria para dominar la potencia de Linux. El libro tiene un nivel adecuado para todos aquellos familiarizados con el mundo de los sistemas operativos aunque no tengan conocimientos previos en entornos Unix/Linux. En el interior del CD-ROM que acompaña a el libro se encuentra la versión Slackware 3.0, (versión que entregó Sólo Programadores en su número 18), código fuente completo y utilidades adicionales.



Editorial: Prentice Hall
Autores: Tacket & Gunter
845 páginas y CD-ROM
Idioma : Castellano

¡SORTEO DE 50 CONEXIONES A INTERNET!

RELLENA ESTA ENCUESTA Y PARTICIPA EN EL SORTEO

SECCIONES	VALÓRALAS DE 1 A 10
NOTICIAS Y LIBROS	1 2 3 4 5 6 7 8 9 10
CARTAS DEL ILUSO	1 2 3 4 5 6 7 8 9 10
TECNOLOGÍA WEB	1 2 3 4 5 6 7 8 9 10
CURSO DE PROGRAMACIÓN	1 2 3 4 5 6 7 8 9 10
SISTEMAS ABIERTOS	1 2 3 4 5 6 7 8 9 10
GRANDES SISTEMAS	1 2 3 4 5 6 7 8 9 10
CURSO DE REDES	1 2 3 4 5 6 7 8 9 10
SIST. OPERATIVO LINUX	1 2 3 4 5 6 7 8 9 10
CÓMO PROG. UNA DEMO	1 2 3 4 5 6 7 8 9 10
MODOS X DE LA VGA	1 2 3 4 5 6 7 8 9 10
PC INTERNO	1 2 3 4 5 6 7 8 9 10
CURSO VISUAL BASIC	1 2 3 4 5 6 7 8 9 10
CURSO VISUAL C++	1 2 3 4 5 6 7 8 9 10

¿Qué te parece el nivel de los artículos?

☐ Muy alto ☐ Alto
☐ Normal ☐ Bajo
☐ ....

¿Qué te parece en general el contenido de la revista?

☐ Muy bueno ☐ Bueno
☐ Regular ☐ Malo
☐ ....

¿Qué cambios te gustaría realizar? ¿Cúales?

☐ AÑADIR NUEVAS SECCIONES ________ ________ ________
☐ QUITAR SECCIONES ________ ________ ________
☐ AMPLIAR LAS ACTUALES ________ ________ ________

¿Te conectas a algún servicio On-Line?

☐ Sí ☐ No
☐ Habitualmente ☐ Esporádicamente

¿Cúal?

☐ INTERNET
☐ INFOVÍA
☐ COMPUSERVE
☐ BBS
☐ OTROS

DATOS PERSONALES

Nombre y Apellidos ____________________
Domicilio ____________________
Población ____________________
C.P. ____________________
Provincia ____________________
Teléfono ____________________
Edad ____________________
Estudios ____________________
Profesión ____________________
Ordenador de que dispone ____________________

Configuración de tu equipo:

Procesador	386	486	PENTIUM < 120	PENTIUM > 120		
Tarjeta Gráfica	VGA	SVGA	65K COLORES	16,7M COLORES		
Tarjeta de Sonido	SOUND BLASTER	SOUND BLASTER 16	GRAVIS ULTRA			
CD-ROM	1X	2X	4X	6X	8X	
Módem	2.400	9.600	14.400	28.800		

RELLENA EL CUESTIONARIO Y ENVÍALO A:

SÓLO PROGRAMADORES (Tower Communications)
Sorteo Internet
C/ Aragoneses, 7 - 28100 Pol. Ind. Alcobendas (Madrid)

Tu revista favorita y la empresa Datagrama te ofrecen la posibilidad de participar en un sorteo de 50 cuentas de conexión a Internet. Para participar, sólo tienes que rellenar los datos que aparecen en esta página y enviarla (sirve fotocopia) a la dirección que aparece unas líneas más arriba.
Nota: Las conexiones son efectivas durante un mes sin limitación diaria.

ACTUALIZACIÓN DINÁMICA DE DOCUMENTOS

Fernando J. Echevarrieta

Aunque el WWW nació como un servicio de hipertexto distribuido con el objeto de compartir información, pronto se convirtió en la principal interfaz de acceso a Internet y, por ello, ha sido necesario ampliarlo constantemente en la búsqueda de una mayor interactividad. En el presente artículo se expone una de las técnicas más simples que permiten la presentación de documentos dinámicos.

Probablemente Berners Lee nunca imaginó cuando pensaba en un mecanismo hipertexto distribuido que le permitiera compartir información con sus colegas del CERN, que su idea, el WWW, llegaría a convertirse en la interfaz "definitiva" de acceso a Internet. La idea original consistía simplemente en permitir el acceso a ciertos documentos sin necesidad de saber donde se encontraban ni tener que realizar conexiones explícitas para conseguirlos. Sin embargo, una idea tan simple, resultó ser de enorme utilidad. Así, el sistema se extendió como una reacción en cadena llegando hasta los últimos rincones del ciberespacio, aunque esta es una historia que ya todos conocemos. El caso es que, a medida que la tela de araña iba cubriéndolo todo, surgían los propósitos de incluir nuevos servicios en la misma.

NECESIDAD DE UNA ACTUALIZACÓN DINÁMICA

El problema fundamental que presentaba el WWW era la interactividad, término que ha marcado como ningún otro las directrices en la evolución del servicio. Así, en sus orígenes, la interactividad era prácticamente nula. Se limitaba a elegir cuál era el siguiente ítem de información a recuperar. La información era algo estático, permanente, que debía cambiarse manualmente. Con la aparición de los forms y la interfaz CGI, se hizo posible un mayor control a la hora de filtrar la información que se deseaba recibir así como, lo que es más importante, se permitía que esta información se generara en el momento de realizar la consulta, ya que la información mostrada era, en realidad, un documento generado dinámicamente en el momento. Pero una vez lograda la *generación dinámica* de información, se trató de avanzar un paso más en la búsqueda de una *actualización dinámica* de esa información.

Hasta el momento, para que la información que aparecía en pantalla fuera actualizada, era necesaria una intervención humana que mediante una acción, por ejemplo, volver a cargar página o realizar una nueva solicitud, generara una conexión que recogiera los nuevos datos. De esta forma, era del todo imposible realizar una monitorización de cualquier sistema. Por poner un ejemplo, si se deseaba llevar un seguimiento de cuántas personas se encontraban conectadas a una máquina así como sus entradas y salidas, era necesario tener sentado frente al ordenador a un operador que, cada cierto tiempo, pulsara el enlace correspondiente para acceder al hipotético CGI que facilitaría esa información. Esto, aparte de incómodo, presentaba ciertos problemas ya que se daba la posibilidad de que entre una y otra solicitud del usuario entraran y/o salieran n mil personas del sistema o, en el otro extremo, se podría dar el caso en que el usuario se cansara de pedir una y otra vez el documento para recibir siempre la misma información debido a que el sistema no hubiera sufrido cambios.

Así pues, el WWW, la interfaz "definitiva" de acceso a Internet no servía para algo tan simple como la monitorización de los sistemas más sencillos.

LA BÚSQUEDA DE SOLUCIONES

El problema de la actualización dinámica de documentos no era únicamente un capricho, sino una necesidad real. Hablando en términos de mercado, no sólo podría ser interesante la monitorización de ciertos tipos de evento más o menos científicos o académicos, como el ejemplo citado de personas conectadas a un sistema, sino que, desde un punto de vista más cercano al usuario final, podría permitirse la consulta de páginas que ofrecieran información continuamente cambiante, como las cotizaciones de bolsa, llegadas y salidas de los aeropuertos, o monitorización de las vídeo-cámaras de control de tráfico. Todos estos ejemplos se emplean hoy en día en Internet. Algunos, incluso, pensamos que aunando la tecnología existente de forms y CGIs que permitían el envío de información a un servidor, con las soluciones de actualización dinámica de documentos, sería posible realizar conferencias múltiples entre usuarios, a modo de chat, talk, o IRC integrados en el WWW. El autor dispone de un ejemplo inacabado aunque operativo de WebTalk en su servidor de WWW, que invita al lector a visitar. Hoy en día se ofrece gratuitamente un sistema prácticamente idéntico pero desarrollado completamente en *http://www.geocities.com*.

Para alcanzar estos objetivos, y sabiendo que existía un potencial mercado, los desarrolladores de browsers, por aquel entonces Netscape y Mosaic (Internet Explorer era aún sólo un anuncio), presentaron sus propuestas con mayor y menor acogida respectivamente.

Mosaic presentó una especificación de interfaz para el control de clientes denominada *CCI* (*Common Client Interface*). Se trataba de una serie de normas para los clientes, del mismo modo que CGI lo hacía para servidores. Así, un programa CGI que también cumpliera la especificación CCI podría realizar una conexión con el cliente para controlarlo directamente e indicarle cuando actualizar la información (entre otras cosas). Claro, que el usuario debía configurar su cliente indicando un puerto de comunicaciones por el que atender peticiones y... En definitiva, demasiado complejo, no sólo para el desarrollador sino también para el usuario, por lo que en este momento descansa en paz en el mítico "cementerio de los estándares perdidos".

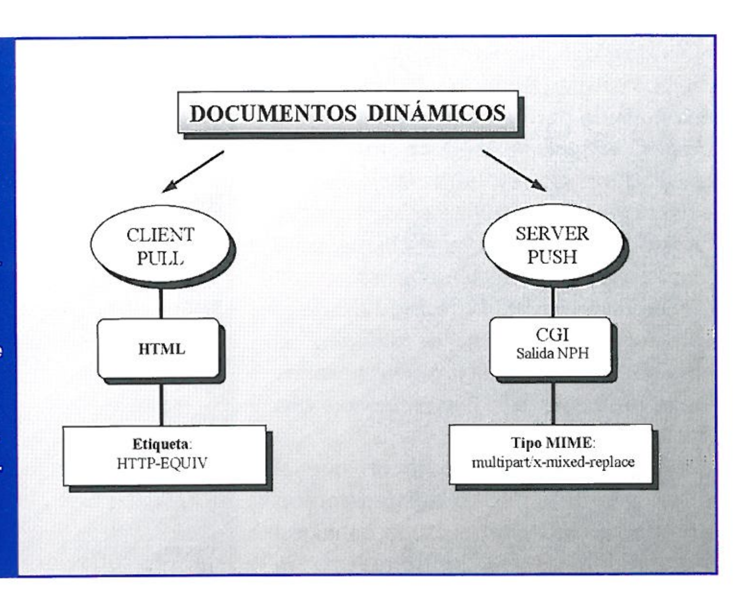


Figura 1. Propuestas de Netscape para la actualización dinámica de documentos. El client-pull, en el lado del cliente se fundamenta en una etiqueta HTML mientras que server-push, en el lado del servidor, se basa en un nuevo tipo MIME generado por un CGI NPH.

Netscape, por su parte, incorporó con mayor fortuna dos nuevas propuestas basadas en estándares abiertos a su browser desde la versión 1.1: el *client-pull* y el *server-push*, siendo este último el sistema sobre el que se centrará el resto del artículo.

Posteriormente, surgieron todas las nuevas tecnologías basadas en módulos software como los *plug-ins* y *Java* . Los *plug-ins* (introducidos en las versiones 2.x de Netscape), como CCI, se basan en la idea de dotar al cliente de una interfaz, esta vez de programación, una API. De este modo, es posible ampliar su funcionalidad mediante la incrustación de nuevos módulos software. Java (experimental con la interfaz alfa en HotJava e introducido al mercado con la interfaz beta y definitiva con Netscape 2.0), como se ha tratado con mayor profundidad en anteriores artículos [Echeva-1] y [Echeva-2], ha hecho posible el desarrollo de arquitecturas *Network Centric*. Se trata de sistemas de desarrollo de propósito general prácticamente transparentes al usuario (como debe ser si se espera tener éxito con una nueva tecnología) en los que lograr una actualización dinámica es una más de sus potenciales funcionalidades. Sin embargo, son infinitamente más complejos para el desarrollador (y no siempre más eficientes) que las técnicas de actualización dinámica que vamos a tratar.

EL "PULSADOR AUTOMÁTICO": CLIENT-PULL

La primera de las soluciones propuestas por Netscape para la actualización dinámica viene del lado del cliente, a través de la etiqueta *<META>* de HTML 3.0, que se trató en su momento en [Echeva-3]. Como rápido recordatorio se mencionará que esta etiqueta debe ser utilizada antes que cualquier otra cosa en el documento y permite realizar documentos que transcurrido un tiempo en *segundos*, carguen el contenido de un determinado *URL*. La sintaxis es la siguiente:

<META HTTP-EQUIV="Refresh" CONTENT="<segundos>; [URL=<URL>]">

si no se indica *URL*, el efecto será el de volver a cargar el propio documento transcurrido el tiempo indicado. Esto alcanza su sentido, bien a través de la carga de una secuencia de URLs, a modo de presentación, bien a través de la carga repetida de un mismo documento que cambie con el tiempo y vaya reflejando la información que se desea monitorizar, por ejemplo, un CGI. Con ello, ya se evita que el usuario tenga que pulsar ningún botón para acceder a información actualizada. Si el lector desea ampliar información acuda al artículo indicado.

LA ESTRELLA: SERVER-PUSH

El uso de *client-pull* es una solución simple y eficiente, sin embargo, no permite un control adecuado de la actualización de la información, al verse ésta "muestreada" tras un intervalo temporal que debe fijarse de antemano. La solu-

ción ideal debería permitir que la información se actualizara en "tiempo real", entendiendo por tal al momento "más o menos" en que se produce un cambio en la información. Para ello, debería existir algún mecanismo de control directo desde el servidor al cliente, para lo que se propuso el *server-push*.

Como se ha venido citando desde el comienzo de esta serie de artículos, el empleo de tecnologías WWW basadas en el protocolo HTTP cuenta con dos *handicaps*:

Por una parte, el hecho de que el HTTP sea un protocolo de transmisión sin estado, es decir, que cada conexión sea como la primera que se realiza, sin que exista la posibilidad de mantener una memoria de lo que se ha realizado hasta el momento.

Por otra, el hecho de que para recuperar cada ítem de información sea necesario realizar una conexión y que cada conexión devuelva un solo ítem de información. Y, lo que es peor, si se emplea un CGI, hasta que este muere, esta información no aparece en la pantalla del cliente (figura 2).

Para lograr una actualización en "tiempo real" del documento mostrado al cliente, sería necesario poder evitar esta segunda restricción, de tal forma que la representación en pantalla del cliente quedara bajo control del servidor.

Así, el servidor, a través de un CGI, enviaría un ítem de información, por ejemplo, las cotizaciones de bolsa en un instante, un fotograma recogido de una cámara de vídeo, quién entra a una máquina y cuántos usuarios hay en ella o qué mensaje nos envía nuestro interlocutor. El cliente, al recibir la información, la mostraría en pantalla y no cerraría la conexión HTTP (¡trampa!), esperando recibir el siguiente ítem de información cuando ésta cambiara (¡más trampa!). El CGI, que no se habría "muerto" (¡otra trampa!) podría enviar otro ítem de información (¡trampa!, ¡trampa!) cuando lo deseara (por ejemplo, un cambio en una cotización de bolsa, un nuevo fotograma, una entrada o salida de la máquina, o un nuevo mensaje). De este modo, con *una conexión HTTP permanentemente abierta, un CGI "inmortal"* podría ir actualizando la información que aparece en pantalla. El CGI iría "empujando" la información desde el servidor, por lo que el sistema recibe el nombre de *server-push*.

Figura 2. En la ejecución de un CGI "habitual", por cada conexión se puede recuperar un único ítem de información. El diálogo no es directo entre CGI y cliente y la aparición de la información en pantalla supone la "muerte" del CGI.

1. CLIENTE WWW → SERVIDOR WWW
2. CLIENTE WWW .... SERVIDOR WWW → CGI
3. CLIENTE WWW .... SERVIDOR WWW ← CGI
4. CLIENTE WWW ← SERVIDOR WWW

La idea (genial) de Netscape fue no cumplir los estándares cumpliendo los estándares. El planteamiento: "hecha la ley, ¿cómo encontrar la trampa que permita librarse de esta restricción?".

La ley dice: "una conexión por ítem de información". Bien, invéntese un ítem de información que se pueda dividir en varios ítems de información *:-)* y cada vez que se recibe un "trocito", se muestra. El cliente es listo porque le han contado lo del *server-push* y colabora, pero el HTTP que es menos listo, cree que está transportando un sólo ítem porque sólo ve la envoltura.

LOS TIPOS MIME MULTIPART

¿Cómo hacer esto? Toda la información que circula por el WWW se encuentra asociada a un tipo MIME que indica su naturaleza. Rebuscando "por ahí" en los tipos MIME registrados encontramos uno muy interesante denominado *multipart/mixed* que se emplea como envoltura para mensajes que contienen distintos tipos de información. El ejemplo más conocido es el de los mensajes de correo y news que contienen fotos además de texto. Un posible mensaje correspondiente al tipo *multipart/mixed* que contuviera texto y una foto tendría un aspecto como el que aparece en el listado 1:

LISTADO 1

```
Content-type: multipart/mixed;boundary=—HastaLuegoLucas

—HastaLuegoLucas
Content-type: text/plain

[Datos del primer item, en este caso en formato ASCII]

—HastaLuegoLucas
Content-type: image/gif

[Datos del segundo item, en este caso una imagen GIF]

—HastaLuegoLucas—
```

Como se puede apreciar, el tipo MIME en cuestión es una envoltura que deja "contento" al HTTP, aunque dentro contiene tantos ítems de información como se desee y cada uno del tipo que se desee. Por supuesto será necesario indicar el tipo MIME de cada parte con su correspondiente declaración *Content-type* y, como siempre, tras ella habrá que enviar dos retornos de carro.

Para que se sepa dónde comienza cada parte, es necesario especificar una cadena delimitadora de libre elección. Esta cadena se indica en la declaración del tipo multipart mediante la palabra *boundary* (frontera, límite) seguida del signo igual (=) y separada del nombre del tipo por un punto y coma (;). No basta con buscar un "Content-type:" ya que esta cadena podría ser parte del texto. Del mismo modo, aunque es menos probable que la cadena delimitadora aparezca en el texto (ya que siempre se puede elegir una muy, muy rara), ésta debe aparecer siempre en una línea aparte, separada del cuerpo de la parte anterior.



Cuando no queden más partes, se vuelve a indicar la cadena delimitadora, esta vez finalizada en dos guiones, lo que indica el final, no sólo de la última parte sino de toda la envoltura.

LA TRAMPA

Hasta aquí, el estándar, la norma. Pero esto es insuficiente para lograr una actualización dinámica. Cualquier lector que se haya apresurado a emplear este tipo MIME comprobará que el resultado es el mismo que presenta cualquier lector de news. La información aparece una a continuación de la otra, en lugar de sustituirse. Es decir, los fotogramas de vídeo aparecen uno a continuación del otro y no uno sobre otro, los textos se acumulan... y lo que es peor, hasta que muere el CGI el cliente no muestra nada. ¿Qué es lo que falla? Pues que aún no se ha hecho "la trampa". Y la trampa, que no es tal, consiste en "sacarse de la manga" un nuevo tipo MIME experimental (lo que es perfectamente lícito, ya que hay un organismo que regula los registros de los nuevos tipos MIME, que para eso están) denominado
multipart/x-mixed-replace

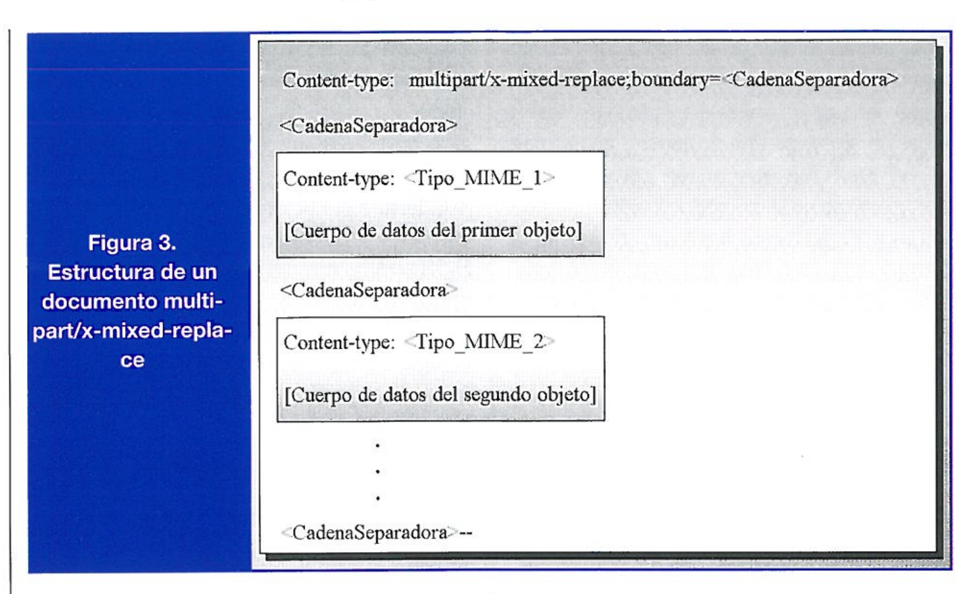
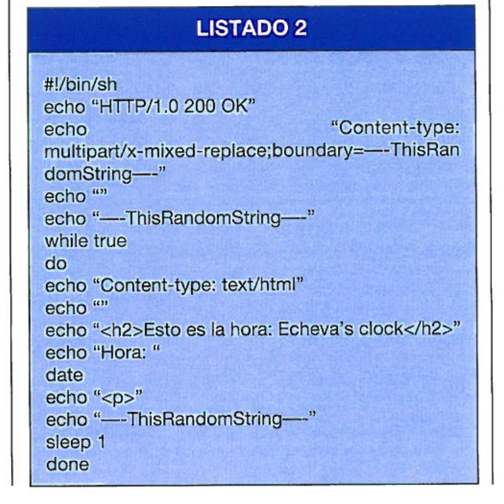


Figura 3. Estructura de un documento multipart/x-mixed-replace

Este tipo MIME emplea una sintaxis idéntica al *multipart/mixed* que se acaba de estudiar. La "*x*" indica que es experimental y la palabra "*replace*" indica que la información de cada bloque debe reemplazar a la del anterior bloque. Así un mensaje reemplaza a otro, una tabla de cotizaciones reemplaza a otra y un fotograma reemplaza a otro, aunque también es posible que una imagen reemplace a un texto y todas las posibles combinaciones que se nos puedan ocurrir, lo que permite el control en tiempo real de presentaciones accesibles a través de WWW. En la figura 3 puede apreciarse la estructura de un documento correspondiente a este tipo MIME.

Así pues, para crear un documento de actualización dinámica en tiempo real es necesario tener presente los siguientes puntos:

1. La conexión estará controlada por un script CGI.
2. Este CGI muestra por su salida estándar un documento del tipo *multipar/x-mixed-replace* y enviará un bloque de información cada vez que desee actualizar la pantalla.
3. Con el objeto de que el CGI actúe sobre el cliente directamente, debe tratarse de un CGI de *salida NPH* (figura 4), aspecto en el que nos detendremos en el siguiente apartado.
4. Como la conexión HTTP permanecerá siempre abierta, no es necesario que este documento termine nunca. La información se irá actualizando en pantalla con cada bloque que el CGI envíe y el cliente permanecerá a la espera indefinidamente siempre que el documento no termine (figura 4). La conexión se cerrará si el usuario pulsa el botón de STOP de su browser, si accede a otro documento al pulsar sobre un enlace, si el CGI finaliza el documento al enviar la cadena delimitadora seguida de dos guiones o, simplemente, si muere.
5. Se considera finalizado un bloque y se muestra la información cuando se encuentra la cadena delimitadora del siguiente bloque. Si esta cadena termina en dos guiones, la conexión se cerrará, si no, el browser seguirá a la espera.



LISTADO 2

```
#!/bin/sh
echo "HTTP/1.0 200 OK"
echo                                   "Content-type:
multipart/x-mixed-replace;boundary=—-ThisRan
domString—-"
echo ""
echo "—-ThisRandomString—-"
while true
do
echo "Content-type: text/html"
echo ""
echo "<h2>Esto es la hora: Echeva's clock</h2>"
echo "Hora: "
date
echo "<p>"
echo "—-ThisRandomString—-"
sleep 1
done
```

Nota importante: algunos servidores de WWW no aceptan espacios en blanco en el "Content-type". Por ello, conviene no incluirlos jamás en la cabecera. Por ejemplo,
Content-type: multipart/x-mixed-replace; boundary=HastaLuegoLucas
podría fallar en algunos servidores, por lo que habría que emplear:
Content-type: multipart/x-mixed-replace;boundary=HastaLuegoLucas

CGIs CON SALIDA NPH

Como se comentó someramente en el artículo dedicado a la descripción avanzada de la interfaz CGI [Echeva-4], existen dos tipos de scripts CGI, los de *salida PH* (Parse Headers) y los de *salida NPH* (No Parse Headers). Los *PH*, son los habituales con los que se ha estado trabajando, mientras que los *NPH* deben devolver un mensaje HTTP completo. El servidor enviará este mensaje completo al cliente sin realizar ningún tipo de análisis de las cabeceras (No Parse Headers) y el buffering intermedio realizado por el servidor se reduce a la mínima expresión, por lo que, en la práctica, es como si el CGI "hablara" directamente con el cliente (figura 4).

Dado que no se va a descender al nivel del protocolo, bastará con saber que es suficiente con incluir el mensaje "*HTTP/1.0 200 OK*" inmediatamente antes de la cabecera *Content-type* del

CGI para convertir un script CGI PH en un NPH. Así pues, en el listado 2 se puede observar un primer ejemplo de script NPH en shell que hace uso de la técnica *server-push* (lo que suena muy "atómico" pero al final se reduce a un par de líneas). El script se limita a actualizar la fecha y la hora cada segundo en la pantalla del browser (figura 5). Al ejecutarlo, pronto se comprueba que en algunas ocasiones el reloj salta dos segundos y cuanto más lejos esté el cliente del servidor, más grandes serán los saltos. Se trata simplemente de un problema productor/consumidor, en el que la transmisión por la red supone un tiempo mayor al período de actualización de la información. Es importante tener esto en cuenta a la hora de definir los intervalos mínimos de actualización.

Nota importante: algunos servidores deben saber que un script es NPH antes de ejecutarlo, por lo que es necesario nombrarlos de manera especial. Así, el servidor del CERN, facilitado con [Echeva-5], requiere que los nombres de los scripts NPH, comiencen con la cadena "nph-". Por ejemplo "nph-reloj".

SH Y SERVER-PUSH

No es recomendable utilizar sh como lenguaje de programación para CGIs que hagan uso de *server-push* en bucles infinitos, como el que aparece en el listado 2. Esto se debe a que si el cliente interrumpe la conexión, el programa en sh nunca se enterará y, por tanto, permanecerá vivo para siempre consumiendo recursos. Un programa en shell no es problema, pero si mil personas hacen uso de un CGI en shell, se tendrán 1000 programas haciendo uso de los recursos de forma inútil. Por ello, en caso de emplear *server-push*, es recomendable utilizar otros lenguajes como Perl o C.

LA LIBRERÍA SVR_PSH

El empleo de la técnica *server-push*, como se puede comprobar, no representa ningún tipo especial de concepto. Se trata más bien de seguir una serie de pasos de manera mecánica. Sin embargo, el sistema es muy sensible a los espacios en blanco y caracteres no visibles que pudiera teclear el programador, por lo que es frecuente encontrar el caso en que se copia código impreso que ha sido verificado y funciona correctamente y al compilarlo y ejecutarlo, el programa no funciona. Por ello, el autor ha desarrollado una sencilla librería C que permite que cualquier programador haga uso de esta técnica sin necesidad de conocerla siquiera. Esta librería, así como la documentación que a continuación se incluye, se distribuye bajo licencia GNU Library General Public License, lo que la califica como "free software" sujeto a las restricciones de la licencia que figuran en el disco de la revista en el fichero LGPL.HTM.

La librería define 3 funciones:

int init_multipart();

Esta función se debe emplear al comienzo del CGI ya que es la encargada de enviar las cabeceras HTTP necesarias en todo CGI con salida NPH así como de indicar que el programa va a utilizar un tipo MIME multipart/x-mixed-replace. Al ser invocada, comprueba la variable de entorno HTTP_USER_AGENT con el fin de verificar si el usuario emplea un browser Netscape. Si es así, envía las cabeceras mencionadas y, en caso contrario genera un mensaje de error y termina.

*int multipart(int time, char *mime_type, int (*funcion)());*

Esta es la función a emplear cada vez que se desee realizar una actualización de la información en pantalla del cliente. Admite tres parámetros: el primero, *time*, determina el valor en segundos que esperará la función antes de enviar el bloque de información; el segundo, *mime_type*, indica el tipo MIME al que corresponderá la información; y el tercero, *función*, es el nombre de la función encargada de generar por la salida estándar la información que se mostrará en la pantalla del browser. La única restricción que tiene esta función es que lo último que debe generar es un retorno de carro (que será el encargado de separar la cadena delimitadora y en la próxima versión de la librería será generado por la propia librería).

EJEMPLO: Si el programador desea mostrar una imagen en formato gif generada por una función *muestra_gif* (que él ha programado), un segundo después de llamar a la función, debería emplear una llamada de la forma:

multipart (1, "image/gif", muestra_gif)

Así, por ejemplo, para mostrar una animación o un vídeo, bastaría colocar esta llamada dentro de un bucle y colocando una variable en el parámetro time se podría controlar el tiempo de transición entre fotogramas. La función *muestra_gif*, realizada por el programador, debería ser capaz de saber qué imagen mostrar en cada caso por la salida estándar. Esto, se podría controlar mediante una variable global, por ejemplo, o programando la función para que entendiera un número arbitrario de parámetros que podría recibir. Pero con objeto de simplificar, para los casos en que se desee pasar parámetros a esta función se ha optado por añadir una función a la librería para un caso particular:

*int multipart_ext(int time, char *mime_type,int (*funcion)(char *),char *parameters);*

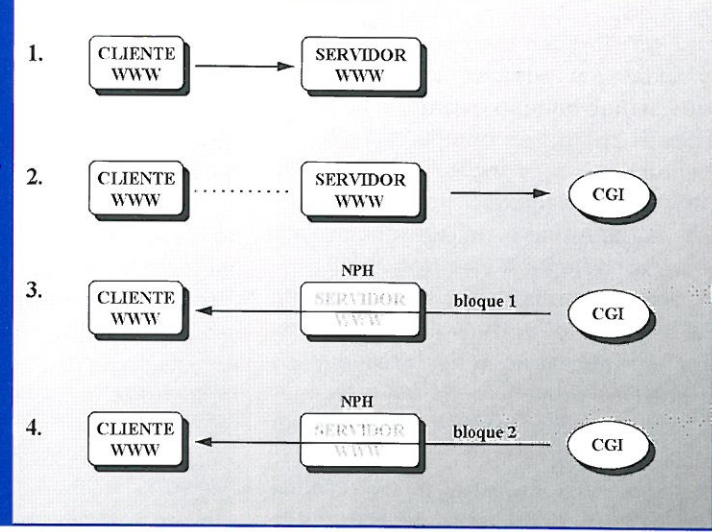


Figura 4. El mecanismo de server-push se logra mediante la utilización de una conexión HTTP permanentemente abierta, controlada por un CGI inmortal con salida NPH que equivale a una conexión directa entre CGI y cliente.

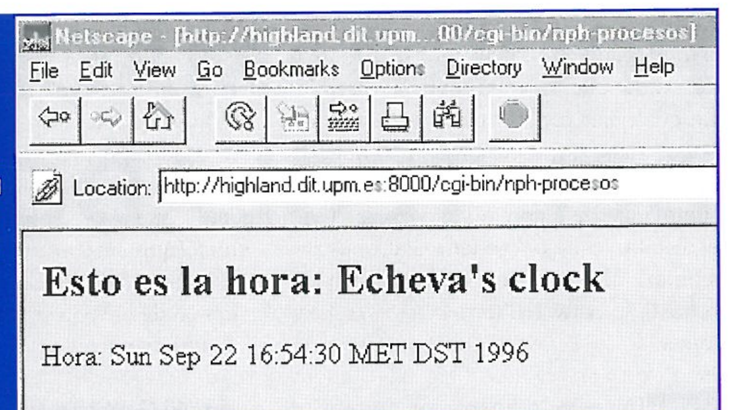


Figura 5. Pantalla correspondiente a la salida del script que figura en el listado 2. La hora se actualiza cada segundo.

LISTADO 3

```
/* nph-etalk.c - NPH CGI receiver */

/*
Modulo: nph-etalk.c
Fecha: Feb 1996
Autor: Fernando J. Echevarrieta Fernandez
Version: 1.0
*/

#include <stdio.h>
#include <string.h>
#include <math.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#include "svr_psh.h"

int port;
int s, c;
struct sockaddr_in mineserver;
char buf[256];

/* */

WebWrite()
{
        buf[c]='\0';
        printf("%s",buf);
}

Welcome()
{
printf("Welcome to / Bienvenido a: <B>Echeva's WebTalk</B><p>");
printf("<B>Echeva's WebTalk</B>. Copyright (c) \
<a href=http://highland.dit.upm.es:8000>
Fernando J. Echevarrieta</A> 1996<p>\n");
}

main(int *argc, char **argv)
{
    port = atoi(argv[1]);

mineserver.sin_family  =AF_INET;
mineserver.sin_port    =htons(port);
mineserver.sin_addr.s_addr=inet_addr("127.0.0.1");

s=socket(AF_INET,SOCK_STREAM,0);

/* Pide conexion al server */
if (connect(s,(struct sockaddr *)&mineserver,sizeof(mineserver)) == -1)
{
perror("client:connect");

        exit(1);
}

write(s,"SOY UN CGI",sizeof("SOY UN CGI"));
read(s,buf,sizeof("SOY UN CGI"));

init_multipart();
        multipart(0,"text/html",Welcome);

        while (1) {
          if ((c=read(s,buf,255)) <=0)
            exit(0);

multipart(0,"text/html",WebWrite);
        }

        close(s);
}
```

Con objeto de permitir el paso de parámetros de forma sencilla a la función encargada de generar la información que actualizará la ventana del cliente, se ha definido la función *multipart_ext*, para el caso particular de las funciones a las que se les pasa una cadena como parámetro. En esta cadena el programador podrá codificar los parámetros de la forma que le sea más cómoda ya que será él, el que defina la función *funcion*. Por ello, ahora *multipart_ext* requiere un parámetro más, que es la cadena que le pasará a la función del programador.

UN SENCILLO EJEMPLO

Como el movimiento se demuestra andando, en el listado 3 se ha incluido el código del CGI responsable de actualizar dinámicamente los mensajes que aparecen en el Echeva's WebTalk (figura 6). Al tratarse de un programa de comunicaciones, la primera parte de la función *main* es código de un cliente TCP que se conecta con el multirrepetidor de mensajes. Esta parte no nos interesa en absoluto ni es necesario entenderla para nuestro caso, pero, para aquellos lectores interesados en el tema, se trató en [Echeva-6], [Echeva-7] y [Echeva-8]. El uso de la librería se muestra en las líneas:

init_multipart();
multipart(0,"text/html",Welcome);
while (1) {
if ((c=read(s,buf,255)) <=0)
exit(0);
multipart(0,"text/html",WebWrite);

Como se puede observar, inicialmente se realiza una llamada a *init_multipart*, que se encargará de enviar las cabeceras necesarias. Tras ello, una primera llamada a *multipart* mostrará en pantalla el resultado de ejecutar la función *Welcome* (definida por el usuario de la librería, potencialmente el lector) y que simplemente genera un mensaje de bienvenida en HTML. Tras ello, comienza un bucle infinito en el que se esperan mensajes del multirrepetidor, *read*. Cuando llega uno, se almacena en la variable global *buf* y se ejecuta de nuevo multipart, esta vez invocando a la función *WebWrite*, que no hace otra cosa que terminarlos como cadena e imprimirlos en la salida estándar. Se desea que el mensaje aparezca en pantalla del cliente tan pronto como se reciba, por lo que se le da un tiempo de espera 0 y como se admite que los mensajes contengan etiquetas HTML se pasa la cadena "text/html" a la función *multipart*. Cada nuevo mensaje que se reciba será enviado de la misma forma e irá reemplazando a los anteriores.

En la figura 6, se puede observar también un frame dedicado a la transmisión de vídeo (que en la figura aparece inactivo). El código del CGI es idéntico, salvo que el servidor al que se conecta le envía fotogramas, pero esto no afecta al código ya que no se analiza el contenido del buffer de recepción.

Pruebe el lector a experimentar con estas funciones y comprobar como su pantalla se actualiza sola.

REFERENCIAS

Todas las referencias corresponden a artículos del autor publicados en Sólo Programadores según:

[Echeva-1], "Java: La Revolución Internet", núm. 22
[Echeva-2], "Applets en Java", núm. 23
[Echeva-3], "El Lenguaje HTML (y III)", núm. 18
[Echeva-4], "La Interfaz CGI: Descripción Avanzada", núm. 25
[Echeva-5], "Instalación de Servidores WWW", núm. 19
[Echeva-6], "La Interfaz Socket (I)", núm. 17
[Echeva-7], "La Interfaz Socket (II)", núm. 18
[Echeva-8], "Servidores Concurrentes", núm. 21

PROGRAMACIÓN DEL TECLADO

José C. Remiro



En este número se tratará de mostrar el funcionamiento del teclado, cómo interacciona con el ordenador, el software, y cómo la ROM-BIOS transmite la información correspondiente a los programas de aplicación.

Cuando un usuario pulsa una tecla, produce una señal que, transformada por un sencillo procesador situado en el teclado, produce un código que identifica de forma unívoca la tecla pulsada.

Este código es transmitido al ordenador, por medio de una línea serie con un tipo de transmisión síncrono, para su tratamiento. Además, este procesador dispone de un *buffer* donde se pueden almacenar las acciones realizadas sobre el teclado, cuando excepcionalmente el ordenador no pueda tratarlas.

Dependiendo del tipo de ordenador que se trate, se puede encontrar dentro de él un procesador dedicado a recibir esta información. Este es el caso de los AT. Los modelos anteriores no disponen de él y, por lo tanto, es la CPU la que debe tratar dicha información.

No deben confundirse los códigos de exploración con el código ASCII. El código de exploración deberá tratarse con posterioridad para hacerlo corresponder con un elemento del código ASCII. A priori, la información generada al pulsar una tecla no se corresponde con el carácter que etiqueta dicha tecla, siendo una rutina de la ROM-BIOS la encargada de realizar su transformación.

Los lenguajes de programación, a través de sus instrucciones de entrada de datos, ocultan la forma en que funciona el teclado. El presente artículo trata de mostrar las estructuras y las funciones de la ROM-BIOS que lo gestionan.

CUADRO 1

Registro AL
byte de estado del teclado utilizando la función del teclado 02h
(1=activo)

7 6 5 4 3 2 1 0

- 0: Mayúsculas derecha pulsada
- 1: Mayúsculas izquierda pulsada
- 2: Tecla Ctrl pulsada
- 3: Tecla Alt pulsada
- 4: Bloqueo de desplazamiento activo
- 5: Bloqueo numérico activo
- 6: Bloqueo de mayúsculas activo
- 7: Inserción activa

Un código de exploración provoca una interrupción hardware (IRQ 1). Ésta a su vez provoca la llamada a la interrupción 09h, donde se encuentra una rutina de la ROM-BIOS que transforma los códigos de exploración (los bytes de datos del teclado se encuentran en el puerto de E/S 60h) en información significativa para el software, es decir, en código ASCII. En principio, esta transformación no es sencilla, pues el código ASCII correspondiente puede haber sido generado con la pulsación de una serie de teclas y teniendo en cuenta el estado de otras. Además, debido a los diferentes modelos de teclado, los códigos de exploración pueden ser diferentes para cada uno de ellos (la ROM-BIOS de arranque suprime estas diferencias, configurando el teclado para que sean compatibles con el de 83 teclas).

Otro problema asociado es el de los caracteres adoptados, es decir, aquellos caracteres singulares de un determinado idioma, por ejemplo, el carácter "ñ" para el castellano. La rutina de tratamiento del teclado de la ROM-BIOS está preparada para utilizar el juego de caracteres estadounidense. La instalación del programa KEYB, controlador del teclado perteneciente al DOS, permite la transformación adecuada de los códigos.

Por último, el código ASCII que ha sido generado por la pulsación de una o varias teclas no se transmite directamente al programa, sino que se deja en un *buffer*. La labor del controlador del teclado termina aquí, siendo el programa de aplicación, a través de las instrucciones propias del lenguaje utilizado, el que debe recoger la información almacenada en dicho *buffer*.

CUADRO 2

Valores del registro BL en la función 03h. (en caracteres por segundo)

valor	*ratio*	*valor*	*ratio*	*valor*	*ratio*
00h	30	0Bh	10,9	16h	4,3
01h	26,7	0Ch	10	17h	4,0
02h	24,0	0Dh	9,2	18h	3,7
03h	21,8	0Eh	8,6	19h	3,3
04h	20,0	0Fh	8,0	1Ah	3,0
05h	18,5	10h	7,5	1Bh	2,7
06h	17,1	11h	6,7	1Ch	2,5
07h	16,0	12h	6,0	1Dh	2,3
08h	15,0	13h	5,5	1Eh	2,1
09h	13,3	14h	5,0	1Fh	2,0
0Ah	12,0	15h	4,6		

Valores del registro BH en la función 03h. (en milisegundos)

valor	retardo (en msg)
00h	250
01h	500
02h	750
03h	1000

CÓDIGOS DE EXPLORACIÓN

No sólo la pulsación de una tecla produce un código de exploración, también cuando se libera una tecla se produce otro código de exploración distinto. Esto permite poder combinar una serie de ellas con el fin de producir diferentes acciones o caracteres.

El código de exploración ocupa un byte. El rango de los códigos de exploración que se producen al pulsar una tecla se encuentra entre 1 y 127, existiendo una relación entre los códigos producidos al pulsar y liberar una tecla.

El código al liberar una tecla es el mismo que el producido al pulsar la misma, pero el séptimo bit está a uno.

Esto quiere decir que un teclado no puede disponer de más de 128 teclas diferentes y que sumando 80h al código de exploración correspondiente al pulsar una tecla se obtiene el código de exploración que se produce al liberarla. Además, teclas que están etiquetadas de igual forma generan un código de exploración diferente. Así, las dos teclas *Shift* del teclado de 102 teclas producirán diferentes códigos.

LA INTERRUPCIÓN 16H

Las funciones 0 y 1 de la interrupción 16h permiten obtener los códigos ASCII generados por el controlador del teclado de la ROM-BIOS. Además, con la función 2 se puede obtener el estado de las teclas de conmutación (por ejemplo, *BloqNum*) y de control (por ejemplo, *Shift*).

El valor de la función para utilizar esta interrupción debe introducirse en el registro AH. Con la función 00h se recupera en el registro AL el código ASCII del primer carácter que se encuentra en el *buffer* del teclado, y se elimina de éste, devolviendo el control al programa que invocó a la interrup-

ción. En el caso de estar el *buffer* vacío, la ejecución del programa se detiene hasta que el usuario pulse una tecla.

La función 01h sirve para conocer si el *buffer* del teclado se encuentra vacío o no. Esta información se suministra por medio del registro de banderas en la *zero-flag*. Ésta contiene el valor 0 si el *buffer* está vacío y 1 en otro caso.

Además, si en el *buffer* del teclado hay al menos un carácter esperando, en el registro AL se puede encontrar el código correspondiente. Esta función, a diferencia de la función 00h, no espera a que el usuario pulse una tecla, simplemente obtiene los valores anteriormente descritos y a continuación devuelve el control al programa que lo invocó.

Hay que tener en cuenta que la anterior función no es capaz de eliminar el elemento que ha obtenido del *buffer*, si es que lo había. Uno de los principales usos que se puede dar a esta función es el comprobar si el usuario ha pulsado una determinada tecla, pero evitando que el programa se detenga hasta que el usuario pulse una si el *buffer* se encuentra vacío. Por otro lado, si se desea no tener en cuenta un código que ya halla sido obtenido con la función 01h, habrá que utilizar la función 00h para eliminarlo del *buffer* y pasar a tratar el siguiente.

Tanto la función 00h como la función 01h devuelven en el registro AL, o bien un cero en el caso de que el carácter sea especial o el código ASCII para los caracteres ASCII ordinarios. En el registro AH se puede encontrar el identificador del carácter si éste es especial, o el código de exploración que identifica la tecla pulsada.

Mediante la función 02h se puede conocer el estado de ciertas teclas de tipo interruptor y control. Esta información se devuelve en el registro AL. En el cuadro 1 se muestran los bits de estado devueltos al utilizar esta función.

En el teclado de los AT es posible establecer una comunicación bidireccional entre el ordenador y el teclado para ajustar el ratio de repetición de un carácter cuando se mantiene pulsada una tecla y el retardo hasta que el segundo carácter y siguientes son generados. Esta función se conoce como *typematic*, y desde el sistema operativo se puede modificar con el comando *mode con rate=nn delay=nn.*

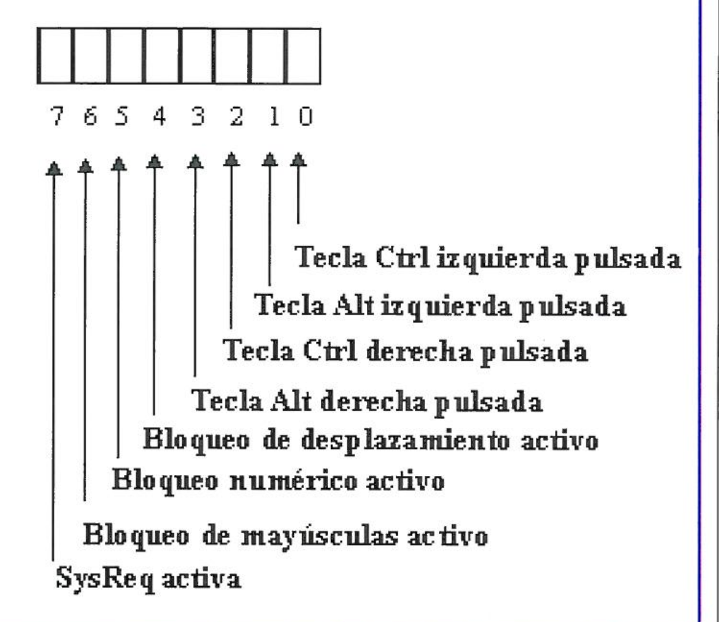
A través de la ROM-BIOS se pueden establecer los anteriores parámetros utilizando la función 03h de la interrupción 16h, con el valor 03h en AH y 05 en AL. En el registro BL debe encontrarse un valor entre 00h y 1Fh, que indicará el ratio. En BH debe establecerse el retardo. En el cuadro 2 se muestran los valores de los registros BL y BH y su correspondencia en caracteres por segundo.

No sólo es posible obtener caracteres del teclado, sino que se pueden introducir en él sin necesidad de que el usuario pulse ninguna tecla, la función 05h se encarga de esto. Basta con suministrar el código ASCII correspondiente en el registro CL y el código de exploración en el registro CL. Esta función preserva los caracteres que ya se encontraban en el *buffer*. Habrá que tener cuidado si se utiliza esta función, pues la capacidad máxima del *buffer* es de 15 caracteres.

Todas las funciones descritas anteriormente sirven tanto para un teclado de 84 teclas como para uno de 101/102 teclas (teclado MFII). Para poder controlar las teclas extra de que dispone el teclado de 101/102 teclas se han incluido las funciones 10h, 11h y 12h. Las dos primeras son equivalentes a las funciones 00h y 01h, pero permiten diferenciar las teclas duplicadas y algunas teclas que no aparecen en el teclado de 84 teclas (por ejemplo, las teclas de función F11 y F12). Las teclas que se han añadido respecto al teclado de 84 teclas pueden distinguirse debido a que en el registro AL no se encontrará el valor 00h sino E0h. De todas formas, si no se desea diferenciar entre teclas que tienen el mismo efecto, se puede asignar el valor 00h a dicho registro y realizar la comprobación correspondiente.

En cuanto a la función 12h, devuelve la misma información en el registro AL que al utilizar la función 02h, pero además en el registro AH se puede encontrar un byte de indicadores de estado de diferentes teclas de control y de conmutación. En el cuadro 3 se muestra la correspondencia entre los diferentes valores y las acciones sobre dichas teclas.

CUADRO 3



CUADRO 4

Estructura de los datos del buffer de teclado

Offset	Almacena	Tamaño
17h	Estado del teclado	1 byte
18h	Estado del teclado ampliado	1 byte
19h	Código ASCII	1 byte
1Ah	Dirección del primer elemento del buffer	1 palabra
1Ch	Último elemento del buffer	1 palabra
1Eh	Buffer del teclado	16 palabras
80h	Dirección del primer elemento del buffer	1 palabra
82h	Dirección del último elemento del buffer	1 palabra

EL *BUFFER* DEL TECLADO

Para gestionar la información que proviene del teclado, la ROM-BIOS proporciona realmente dos interrupciones diferentes que no se deben confundir: la interrupción hardware 09h, que recoge la información que proviene del teclado y la almacena, de forma adecuada, en una zona de memoria baja y la interrupción 16h, que proporciona al DOS y a los programas de aplicación la información anteriormente almacenada.

En el cuadro 4 se pueden observar las zonas de memoria que sirven para gestionar la información perteneciente al teclado. Entre esta información se pueden reconocer los bytes de estado del teclado, que se corresponden con la información devuelta por las funciones 02h y 12h de la interrupción 16h.

En la dirección relativa 1Eh se encuentra el *buffer* del teclado que ocupa 16 palabras (32 bytes). Su función es almacenar los caracteres que han sido introducidos por el teclado y aún no han sido tratados. En la dirección relativa 80h se encuentra un puntero que indica el primer elemento a tratar, mientras que en la dirección 82h se encuentra otro puntero que indica el último elemento introducido. Cuando ambos punteros coinciden, el *buffer* se encuentra vacío. La estructura lógica que presenta el *buffer* es la de una cola circular, estando las posiciones situadas entre el puntero del primer y último elemento ocupadas por los caracteres no tratados. Esta estructura evita tener que trasladar los elementos situados en el *buffer* ante la llegada de un nuevo elemento o bien ante la retirada de un elemento del mismo.

El tamaño del *buffer* se corresponde con 16 palabras, es decir, es capaz de almacenar un máximo de 15 caracteres, pues a pesar de que cada carácter está representado por 2 bytes, se deja siempre una posición libre para poder comprobar si está vacío o no.

En el cuadro 5 se pueden observar dos procedimientos diferentes para vaciar el *buffer* del teclado. Estos son útiles cuando en un programa se solicita una respuesta a un usuario y no se desea que ésta sea apresurada, por ejemplo, hasta que no haya aparecido completamente la pantalla que contiene el texto de la pregunta.

El procedimiento *vacia_buff_teclado_1* hace uso de la interrupción 16h para realizar la anterior tarea. Por medio de la función 01h se comprueba si hay caracteres en el *buffer*. Si es así, se procede a vaciarlo con la función 00h de la interrupción 16h. Es aconsejable que este procedimiento sea llamado antes de presentar la pantalla para realizar la pregunta. Si se desea utilizar este procedimiento en un programa habrá que indicar que se debe utilizar la unidad *Dos*, mediante la instrucción *uses*.

El procedimiento *vacia_buff_teclado2* es más expeditivo. Directamente iguala los punteros que indican el primer y último elemento del *buffer*. Recuérdese que ésta era la condición para comprobar que el buffer estuviera vacío. La estructura *memw* es un array predefinido en TurboPascal para acceder directamente a la memoria. Cada elemento de este array ocupa una palabra y el índice utilizado es la base y el desplazamiento del segmento de la posición de memoria a acceder. En los índices se pueden reconocer las direcciones anteriormente descritas para los punteros del *buffer* del teclado.

Hay que tener en cuenta que se puede producir una interrupción mientras se produce la asignación descrita anteriormente, y que ésta puede utilizar las estructuras de datos del teclado, lo que podría producir una asignación errónea en los punteros. Por esta razón, antes de realizar la asignación se han desactivado todas las interrupciones hardware mediante la instrucción *inline($FA)*, que se corresponde con la instrucción del procesador *CLI* (*clear interrupt flag*). Posteriormente se vuelven a activar con la instrucción *inline($FB)* que se corresponde con la instrucción del procesador *STI* (*set interrupt enable flag*).

EJEMPLOS

En el CD-ROM que acompaña a la revista se puede encontrar la unidad

CUADRO 5

PROCEDIMIENTOS PARA VACIAR EL BUFFER DEL TECLADO

```
procedure vacia_buff_teclado1;
 var
  registros: registers;
 begin
  registros.ah:= $01;
  intr($16, registros);
  if (regs.flags and $0040) = 0
   then
    repeat
     registros.ah:= 0;
     intr($16, registros);
     registros.ah:= 1;
     intr($16, registros);
    until (regs.flags and $0040) <> 0;
 end;
```

```
procedure vacia_buff_teclado2;
 begin
  inline($FA);
  memw[$40:$1A]:= memw[$40:$1C];
  inline($FB);
 end;
```

teclado, junto con una serie de programas que la utilizan. Incluye un conjunto de procedimientos y funciones que utilizan parte de los conocimientos expuestos en el artículo. En este apartado se comentará cómo están construidas y cómo utilizarlas.

La función booleana *teclado_102* comprueba si el teclado del ordenador dispone de 84 o de 102 teclas. Esta información puede ser útil para saber si se encuentran disponibles las funciones 10h, 11h y 12h. En principio, la ROM-BIOS no dispone de ninguna función para determinar este hecho, por lo que se ha recurrido a un pequeño "truco" que sirve para cualquier función de la ROM-BIOS: si se realiza una llamada a una función de la ROM-BIOS que no existe, el número de función y subfunción (contenidos en el registro AX) se devuelven sin modificar. Pero es posible que, si existe la función, el valor 12h sea devuelto en el registro AX. Pero con la información de que se dispone; se sabe que el valor 12h colisionaría con los bytes de estado del teclado ampliado. Por lo tanto, si el registro AX devuelve el valor 12h intacto querrá decir que el teclado es de 84 teclas.

El procedimiento *velocidad_tcl* permite ajustar el ratio y el retardo del teclado a través de la función 03h. Admite como parámetros los anteriores valores. Es aconsejable, si se desea utilizar este procedimiento, validar previamente los valores de los parámetros para que se ajusten a los de la tabla del cuadro 2. Esto es precisamente lo que se ha hecho en el programa *ajusta_teclado* (cuadro 6). Antes de proceder a la llamada del procedimiento *velocidad_tcl* se verifica que se han introducido dos parámetros, que éstos son enteros y que se encuentran en el rango de valores admisibles, en caso contrario, se procede a visualizar un mensaje de cómo utilizar el programa.

El programa *demo_estados*, trata de mostrar cómo se puede modificar el byte de estado del teclado, activando y desactivando las teclas que tienen asociado un led (se puede observar al ejecutar el programa cómo se apaga y enciende el correspondiente led). Hace uso del procedimiento *act_o_desc_tecla*, que admite como parámetros el código de una tecla, que aparece en las constantes definidas en la interface de la *unit*, y un valor booleano. Si éste último es verdadero activa la tecla, desactivándola en caso contrario. Para realizar este proceso se utiliza la función 01h y se accede a través de la dirección absoluta al byte de estado.

COMENTARIO

Los lenguajes de programación de alto nivel proporcionan un conjunto de instrucciones más fáciles de utilizar que la ROM-BIOS. Así, Turbo Pascal posee los procedimientos *ReadKey* y *Keypressed*, que implementan las funciones 00h y 01h de la interrupción 16h de la ROM-BIOS. A menos que se desee tener un control absoluto sobre el teclado, es aconsejable utilizar las instrucciones del lenguaje, pues éstas son más sencillas de utilizar y permiten dar generalidad a los programas, además del ahorro de tiempo que supone el trabajar sobre lo ya programado.

Rellena este cupón y envíalo a: ABETO EDITORIAL C/ Aragoneses, 7. 28100 Pol. Ind. ALCOBENDAS (Madrid)



ENCAMINAMIENTO Y DETECCIÓN DE ERRORES

Fernando J. Echevarrieta

En el presente artículo se completa el estudio del nivel 3 de la arquitectura TCP/IP mediante el análisis del encaminamiento IP y la detección de errores asociada, que se realiza a través del protocolo ICMP. A la vez se expone una serie de conceptos fundamentales para la administración de redes basadas en esta arquitectura.

Continuando con la línea de los últimos artículos de la sección en la que se está analizando el funcionamiento interno de Internet, este mes se completan los aspectos relativos al nivel inter-red, más conocido como nivel de red o, simplemente nivel 3. Este es el nivel más importante ya que es aquí donde el protocolo de red, IP, genera la red virtual que se conoce como Internet.

UN ENCAMINAMIENTO BASADO EN TABLAS

La labor fundamental de un protocolo de nivel 3, como IP, es la de *encaminamiento* o *routing*, sin la cual sería imposible hacer llegar la información a su destino. La función de encaminamiento se reduce, en principio, a decidir el siguiente lugar a donde enviar una datagrama en función de su dirección de destino.

El encaminamiento IP se encuentra basado en tablas denominadas IRTs (*Internet Routing Tables*). El software del protocolo dispone de unas tablas que relacionan posibles destinos con rutas a seguir. Estas tablas se podrían representar como en la figura 1 por dos columnas, una que contiene los posibles destino y otra que contiene los nodos de salida (*gateways*) asociados. Estos gateways, como es lógico, se deben encontrar a un solo salto, es decir, directamente conectados al nodo en que se encuentra la tabla, ya que, en caso contrario, surgiría la paradoja de ¡tener que encontrar un gateway para llegar al gateway!

Cuando un nodo recibe un datagrama, el software de IP comprueba su dirección de destino. Si ésta coincide con la del nodo, lo procesa, desfragmentándolo (si procede) como se explicó en el número anterior, elimina las cabeceras de protocolo y entrega la información al nivel superior. En caso de que el destino no sea el propio nodo, deberá tomar la decisión de a quién "pasarle la bola". Así, lo primero que realiza es comprobar si el destino se encuentra en la misma red mediante la comparación de las máscaras de subred, proceso que se describió en [Echeva-1]. Si así ocurre, lo envía directamente a destino (*encaminamiento directo*). En caso contrario, deberá encontrar otro nodo intermedio en el *camino (encaminamiento indirecto)*. Para ello, realiza una consulta a su tabla de encaminamiento. En ella, primero trata de encontrar la dirección del host o sistema final destino. En caso de no hallarla, trata de buscar la dirección de subred del destino. Si tampoco encuentra ésta, empleará la ruta por defecto. En el improbable caso de que tampoco exista ruta por defecto, descartará el datagrama y generará un mensaje de error ICMP, que se estudiará más adelante en este mismo artículo.

El empleo de direcciones de hosts en las tablas de encaminamiento aparece en raras ocasiones en las que se desea distinguir, por alguna razón especial, ese host del resto de su subred. En la gran mayoría de los casos, únicamente aparecerán direcciones de red y por defecto.

Este sistema de encaminamiento lleva una característica asociada y es que todo el tráfico que vaya a una misma red, seguirá un mismo camino (salvo en las rutas específicas para hosts), lo que (perdone el lector lo obvio de esta reflexión, dada su importancia) supone que no se emplearán varios

caminos a la vez. Otra de las ideas que lleva implícitas es que únicamente el último gateway tendrá conocimiento de si el sistema final se encuentra "vivo" o no. Volveremos más adelante sobre esta cuestión al hablar de ICMP.

RETRANSMISIÓN DEL DATAGRAMA

El siguiente paso será el reenvío del datagrama. Para ello, será necesario encontrar la dirección física asociada a la dirección IP lógica encontrada (para mayor información sobre estos tipos de direcciones acuda a [Echeva-2]). Este problema se puede resolver de tres maneras :

1. En primer lugar, se puede resolver mediante una búsqueda basada en tablas que relacionen direcciones físicas y lógicas. Un ejemplo de estas tablas en UNIX es el fichero */etc/ethers* al que se hacía mención en [Echeva-2].
2. Otro posible mecanismo se basa en una asignación de direcciones físicas a lógicas en función de un determinado algoritmo que permita deducir unas en función de otras. Este enfoque, que apenas se emplea, supone una configuración más compleja en la que no se ha entrado en esta serie (ni se entrará), pero se cita para que el lector tenga constancia de su existencia.
3. El empleo de *ARP broadcasting*, es decir, del protocolo de resolución de direcciones (*Address Resolution Protocol*) que ya fue tratado en [Echeva-2]. Para aquellos lectores que no tengan este artículo a mano, se recuerda que el funcionamiento de este protocolo se basa en el envío por broadcast (es decir a una dirección que indica "todas las máquinas de esta subred") de una petición similar a "que alguien me diga quién tiene esta dirección IP". Según especifica el protocolo, sólo una máquina contestará "¡Yo!" en un mensaje que portará su dirección física. Este mecanismo se combina con una caché en la que figuran las últimas direcciones empleadas, por lo que en las siguientes comunicaciones con estas máquinas, no será necesario volver a preguntar.

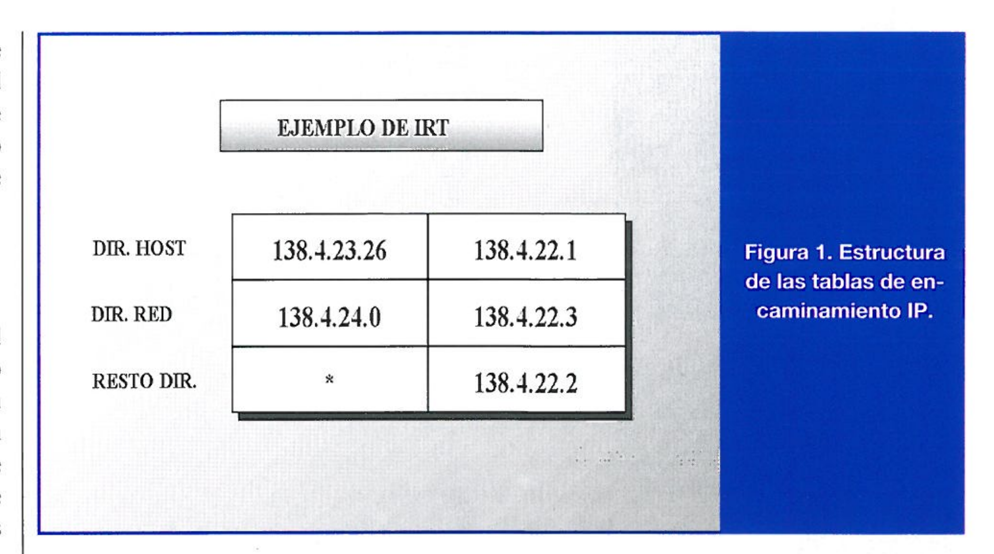


Figura 1. Estructura de las tablas de encaminamiento IP.

Es importante destacar que mientras para la resolución de la dirección física del siguiente nodo se emplea la dirección IP resultante de la función de encaminamiento, la dirección IP destino que figura en el datagrama no se ve alterada ya que, de ser así, el siguiente nodo interpretaría que es su destinatario.

EJEMPLOS DE ENCAMINAMIENTO IP

Como ejemplo, supóngase que una máquina de dirección 138.4.22.23 recibe un datagrama con dirección de destino 138.4.22.25. Si la máscara de subred es 255.255.255.192 [Echeva-1], comprobará que se encuentran directamente conectadas :
Dirección de "mi red"
138.004.022.023
255.255.255.192 &&

138.004.022.000
Dirección de "red destino"
138.004.022.025 &&

138.004.022.000
por lo que la función de encaminamiento dará como siguiente nodo a la propia 138.4.22.25 (encaminamiento directo).

Imagine el lector que ahora se recibe otro datagrama con dirección de destino 138.4.23.36. Siguiendo un proceso análogo, el software IP comprobará que se encuentra en otra subred (138.4.23.0). Por lo tanto no será posible enviarle el datagrama directamente por lo que acudirá a su tabla de encaminamiento. En esta tabla, continuando con el ejemplo de la figura 1, encontrará una dirección de host que coincide con la de destino del datagrama, por lo que la función de encaminamiento dará como resultado el nodo asociado en la tabla, es decir, el 138.4.22.1.

Ante la recepción de un tercer datagrama con dirección destino 138.4.24.15, el software IP nuevamente detectaría que se trata de una máquina a la cual no se encuentra directamente conectado, por lo que debería hacer uso nuevamente de la tabla de encaminamiento. Es esta ocasión, tampoco encontraría en la tabla al destino final, por lo que buscaría su dirección de red : 138.4.24.0. En este caso, sí encontraría una entrada en la tabla para la misma, que daría como siguiente nodo al 138.4.22.3.

El último de los casos, y el más frecuente, se daría cuando el nodo recibiera un datagrama con dirección de destino 142.23.32.42. Una vez más detectaría que no se encuentran directamente conectados. Pero esta vez no encontraría ni la dirección del host destino ni la de la red en su tabla de encaminamiento. Por lo tanto, esta vez, debería acudir a la ruta por defecto, en este caso, a la 138.4.22.2.

Es gracias a la existencia de rutas por defecto, junto al mecanismo de direccionamiento de IP que, como se expuso en [Echeva-1] divide las direcciones en una parte de red y otra de host, a lo que se hace posible el encaminamiento IP. De esta forma, no es necesario que todos los nodos de Internet dispongan de toda la información de encaminamiento de Internet. Esto sería del todo inviable. Por una parte, porque acabaría con la memoria

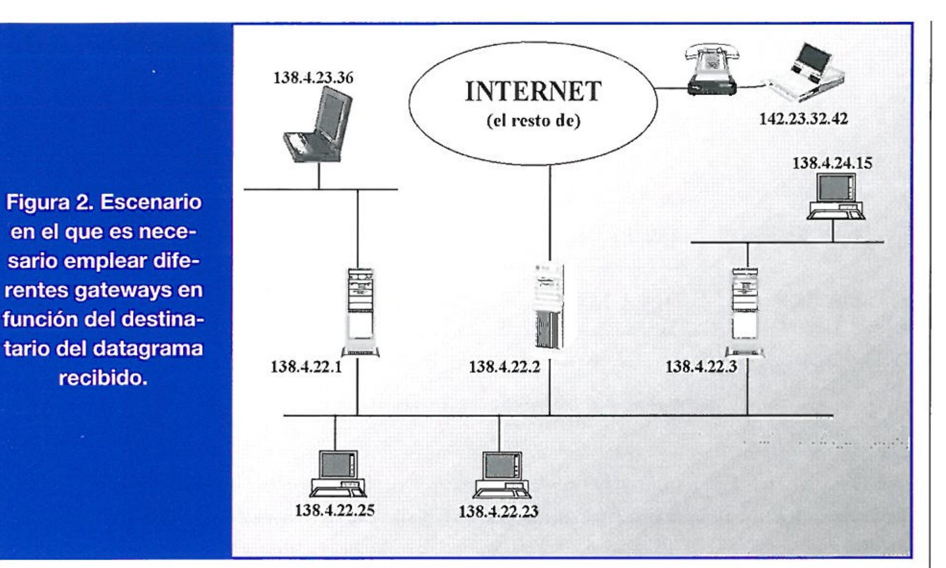


Figura 2. Escenario en el que es necesario emplear diferentes gateways en función del destinatario del datagrama recibido.

de más de un nodo y el recorrido de tan ingente tabla conduciría a una enorme espera, y por otra, porque sería imposible reflejar todas las actualizaciones de algo tan dinámico como Internet, que como un ser vivo se encuentra en constante crecimiento y cambio.

PROTOCOLOS DE ENCAMINAMIENTO

Como se puede comprobar, todo esto dará resultado siempre y cuando se cuente con un conjunto de tablas correcto y consistente. Pero ¿Quién genera estas tablas? La pregunta no tiene solución única. Existen dos formas de mantener estas tablas de encaminamiento. La primera, la más habitual, familiar para todos los administradores de red: *a mano*. En [Echeva-1] se explicó como generar tablas de encaminamiento en UNIX mediante el comando *route*. La segunda forma es a través de *protocolos de encaminamiento*. Los protocolos de encaminamiento deben ser capaces de mantener unas tablas de encaminamiento consistentes en toda la red. Para ello, se mantiene la idea de que en los host y en la mayoría de los gateways se disponga únicamente de información parcial y el resto sean rutas por defecto. La información entre gateways interiores se actualiza de forma consistente a través de protocolos dinámicos denominados IGPs (*Intermediate Gateway Protocols*). Dentro de los IGPs existen dos tipos de protocolos, los de tipo *Vector de Distancias* (que informan del "coste" de cada ruta como posible alternativa), como RIP o HELLO y los basados en *Estado de Enlaces*, como SPF u OSPF.

De esta forma, se divide la Internet en *Sistemas Autónomos*, entendiendo como tales a conjuntos de redes y gateways gestionados por una autoridad administrativa que decide su arquitectura interna de encaminamiento basada en estos IGPs. Pero ahora surge el problema de definir una interfaz de comunicación de la información de encaminamiento entre diferentes Sistemas Autónomos. Esto se realiza a través del EGP (*Exterior Gateway Protocol*). Los gateways "finales" de los Sistemas Autónomos que emplean EGP constituyen el *core* o *backbone* (espina dorsal) de Internet. Por este backbone, o mejor dicho, estos backbones, ya que hace algunos años que Internet dejó de tener una única espina dorsal, circula la información completa de encaminamiento, ya que los gateways envían información de todas las subredes internas. De esta forma, sólo un núcleo central de Internet debe disponer de toda esta información ya que en el backbone no se puede utilizar información por defecto para evitar a toda costa la formación de posibles bucles de encaminamiento.

DETECTANDO ERRORES: ICMP

Como todo programador sabe, los sistemas que funcionan sobre el papel dejan de hacerlo en "el campo de batalla". Y el campo de las comunicaciones es uno de los que más fácilmente le hacen desesperar a uno cuando algo falla, ya que las labores de depuración son especialmente difíciles. Esto se debe a que si, por ejemplo, se envía un mensaje y no se recibe respuesta, podría ser porque el mensaje no se ha llegado a enviar correctamente, lo que, antes o después se descubre con un depurador (o sin él); pero si el error no es local, puede convertirse en una verdadera pesadilla llegar a averiguar qué es lo que realmente ha sucedido. ¿Llegó el mensaje a destino? ¿Existía el destino? ¿Fue el mensaje de vuelta el que se perdió? En el caso de una red física, siempre es posible que el hardware notifique ciertos errores, pero en una red virtual de nivel 3 en las que las máquinas no se encuentran directamente conectadas, este problema debe resolverse por medio del software (de protocolos, claro :-)).

Así, como ya se adelantaba en artículos anteriores, si el protocolo IP es en cierto modo como la sangre de Internet, el protocolo ICMP (*Internet Control Message Protocol*) podría ser considerado el conjunto de impulsos nerviosos que provocan los actos reflejos en un organismo vivo al transportar información de condiciones especiales de la red, en la mayoría de los casos, condiciones de error. También se adelantaba que si bien ICMP es usuario de IP ya que los mensajes ICMP viajan encapsulados en datagramas IP, ambos pertenecen al nivel inter-red y, de hecho, se considera al primero como parte fundamental de este último, por lo que las especificaciones exigen que no pueda darse una implementación del uno sin el otro.

Así, ICMP será capaz de avisar de situaciones anómalas, como la expiración del tiempo de vida de un datagrama, un atasco en las autopistas de la información o el triste fallecimiento del destinatario de los mensajes. Por tanto, la principal función del protocolo ICMP es la de gestión de errores, gestión que se limita a su detección y notificación, pero no a su corrección, que se deja en manos de los protocolos de niveles superiores. Por otra parte, si bien se trata de una valiosa ayuda, los mensajes ICMP se dirigen siempre al remitente del datagrama que se vio involucrado en el suceso, pero ninguno de los nodos intermedios será consciente siquiera de que tuvo lugar tal suceso. Esto se debe a que, como se recordará del capítulo anterior [Echeva-3], los datagramas IP



únicamente contienen las direcciones de remitente y destinatario, pero no hay forma de conocer a priori (salvo si se emplean opciones especiales) las rutas intermedias ni de ida ni de vuelta.

Por otra parte, como se verá a continuación, ICMP presenta algunas deficiencias (insalvables dada la naturaleza de la Internet, determinada por IP) ya que, ni será capaz de detectar todos los errores ni, en la mayoría de los casos, de determinar la causa que los produce.

ANATOMÍA DE UN MENSAJE ICMP

Los mensajes IP se distinguen por... precisamente eso, por distinguirse. Es decir, al contrario que los datagramas IP, que son todos iguales [Echeva-3], cada mensaje ICMP tiene un formato distinto. Esto, lejos de hacerlo más complicado, permite enviar en cada mensaje únicamente la información necesaria. En cualquier caso, todo mensaje ICMP tiene un comienzo común a partil del cual se podrá deducir el formato del mismo. Este comienzo consiste en tres campos que se pueden observar en la figura 4 y se detallan a continuación :

TIPO: Este primer campo de 8 bits será el que determinará el tipo de mensaje así como la estructura del mismo.

CÓDIGO: El segundo campo, también de 8 bits, especifica de forma concreta la situación anómala.

CHECKSUM: Un tercer campo de 16 bits contiene información redundante para la certificación de todo el mensaje, no sólo de la cabecera como ocurría con el checksum de IP [Echeva-3].

Además de estos campos, suele ser común (en todos los mensajes que transportan una notificación de error), que se transporte también en algún lugar del cuerpo la cabecera más los 64 primeros bits del cuerpo de datos del datagrama que generó el error. Esto se debe a que gracias al análisis de estos bits, es posible determinar la aplicación que generó el datagrama con objeto de notificarle el error. En cualquier caso, como se hizo el mes pasado con IP, se facilita en el disco de la revista el RFC 792 correspondiente a ICMP, donde el lector interesado podrá encontrar la descripción detallada del protocolo.

MENSAJES Y SITUACIONES

Los tipos de mensaje transportados por el protocolo ICMP aparecen en la tabla 1 y se deben a las siguientes situaciones:

0 Respuesta de Echo
3 Destino Inalcanzable
4 Congestión
5 Cambio de rutas
8 Solicitud de echo
11 Tiempo excedido por un datagrama
12 Parámetro incorrecto
13 Solicitud de marca de tiempo
14 Respuesta de marca de tiempo
17 Solicitud de máscara de subred
18 Respuesta de máscara de subred

- *Confirmación del estado de la red*

En ocasiones existe la fundada sospecha de que "algo hay torcido en el estado de Dinamarca" (*), por lo que se suele emplear una utilidad presente en la mayoría de los sistemas operativos (como Windows 95 y UNIX) como un comando que recibe el nombre de *ping*. Esta utilidad no hace otra cosa que enviar un mensaje ICMP *Echo Request* a un destino para comprobar si éste se encuentra "vivo" y responde con un ICMP *Echo Reply*. En caso afirmativo, se puede asegurar que la máquina local emplea la ruta de salida correcta, que todos los sistemas intermedios, tanto en la ida, como en la vuelta, funcionan correctamente y disponen de rutas correctas y que el destinatario se encuentra preparado (lo que no es poco). Desgraciadamente, en caso de fallo, no será posible determinar cuál o cuáles de estos factores ha sido el causante.

La forma de emplear el comando ping es simple, por ejemplo :
ping highland.dit.upm.es

Se abre un breve paréntesis para comentar otra interesante utilidad denominada *traceroute* (*tracert* en Windows 95), que proporciona información sobre los nodos intermedios además del destino. Su sintaxis es la misma, así que se deja al lector que experimente con ello.

- *Destino inalcanzable*

Existe un gran número de razones relacionadas con el encaminamiento por las que un datagrama puede no ser entregado en destino. Todas estas razones se notifican mediante un mensaje de TIPO 3 concretado con uno de los códigos que aparecen abajo. Este es un caso típico de mensaje ICMP. El mensaje es generado por el nodo intermedio que no ha sido capaz de entregar el datagrama, por ejemplo, por un fallo en las rutas como el que se preveía en el primer apartado del artículo, y es de los que contiene el comienzo del datagrama original. De esta forma, podrá leerse en él la dirección destino, que es inalcanzable. Por regla general las aplicaciones suelen notificar este error siempre al usuario humano.

0 Red Inalcanzable
1 Host Inalcanzable
2 Protocolo no accesible
3 Puerto no accesible
4 Datagrama no fragmentable, no "cabe" por una red
5 Fallo en el encaminamiento en origen

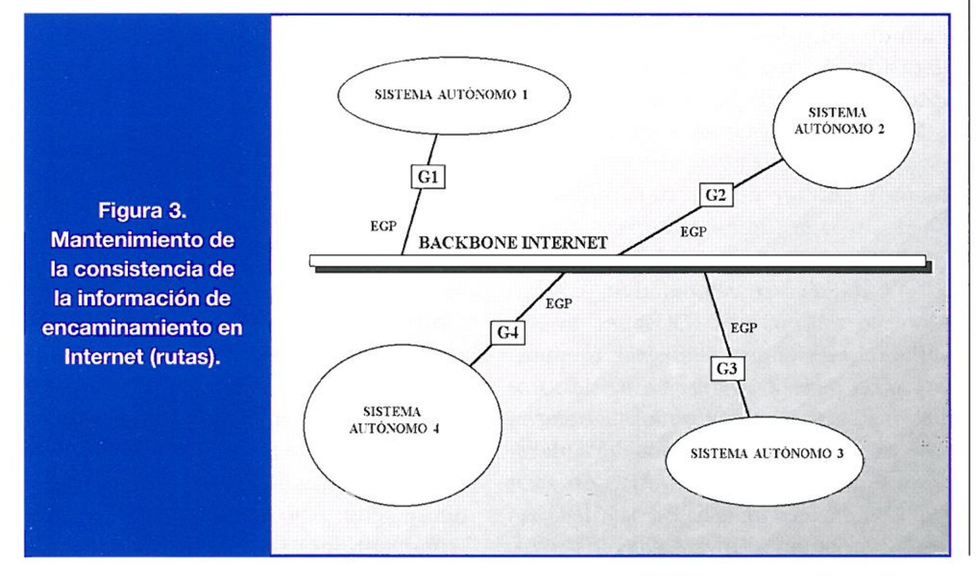


Figura 3. Mantenimiento de la consistencia de la información de encaminamiento en Internet (rutas).

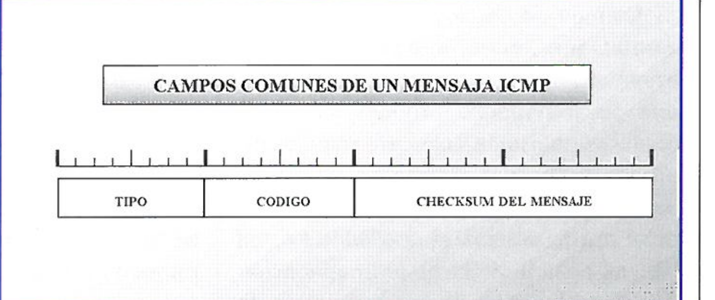


Figura 4. Campos comunes de un mensaje ICMP.

6 Red de destino desconocida
7 Host de destino desconocido
8 La máquina remitente está aislada
9 Comunicación con red destino prohibida por el administrador
10 Comunicación con host destino prohibida por el administrador
11 Red no accesible por el tipo de servicio
12 Host no accesible por el tipo de servicio

- *Rutas no optimizadas*

En otros casos, el error es salvable o se trata simplemente de una "no optimización". Esto ocurre, por ejemplo cuando un gateway recibe un datagrama y comprueba en sus tablas de encaminamiento que habría sido más conveniente enviarlo por otro camino. En estos casos, el datagrama sigue su curso, pero el gateway notifica la corrección de rutas al remitente mediante un mensaje *ICMP redirect.* Se recuerda que no se produce el descarte del datagrama sino que simplemente se trata de una notificación de posible optimización. Esto permite administrar facilmente redes en las que a los sistemas finales se les da una mínima información de encaminamiento, en la practica, un único gateway. En caso de disponer de alternativas mejores irán aprendiendo a base de mensajes ICMP. Sin embargo, también existen lagunas en este mecanismo ya que si el error se debe a una modificación de la red no acompañada de la correspondiente actualización de las tablas de encaminamiento de todos los gateways, el problema no se resolverá, ya que el mensaje ICMP será recibido únicamente por el originario del datagrama mal encaminado y no por el gateway que lo encaminó mal (como se ha justificado anteriormente). Este caso puede observarse en la figura 5, en la que la máquina A envía un datagrama a la máquina Z a través del gateway C. C detecta una "no optimización" de rutas y lo notifica a A, pero esto es inútil debido a que la información incorrecta se encuentra en las tablas de B. Por supuesto, el mensaje ICMP atraviesa B, pero simplemente eso, lo atraviesa y es reenviado sin ser "leído" ya que B no es su destinatario. En cualquier caso, el usuario suele recibir el mensaje en pantalla, por lo que, si lo pone en conocimiento del administrador, tras una revisión podrá corregir las rutas.

- *Congestión en la red*

Otro de los principales problemas, quizá el más habitual, que aparecen en la red, es el de la congestión. Como toda vía, la red se satura cuando el tráfico excede su capacidad. Cuando un nodo recibe más datagramas de los que es capaz de procesar, almacena en un buffer de memoria una cola de datagramas y cuando la cola se llena, comienza a descartarlos. Sin embargo, al igual que esto es posible al tratarse IP de un protocolo no fiable, recuérdese que también se trata de un protocolo, best effort, es decir, que hará lo posible para evitar la pérdida de datagramas [Echeva-3]. Por ello, en situaciones de congestión como la descrita, el nodo congestionado enviará mensajes ICMP *Source Quench* ("atenuación" de la fuente) con el objeto de notificar la situación a la fuente que crea la congestión esperando que ésta "se tranquilice" y reduzca su velocidad de generación de datagramas. Distintas implementaciones eligen diferentes momentos para generar mensajes ICP *Source Quench*, así, algunos nodos generarán uno de estos mensajes por cada datagrama que hayan tenido que descartar por no caber en la cola. Otros, más previsores, comenzarán a enviar los mensajes cuando la cola alcance un punto crítico. En general, una máquina que reciba notificaciones de congestión deberá atenuar su ratio de envío de datagramas por unidad de tiempo y, dado que no existen mensajes ICMP para indicar que se han terminado los problemas de congestión, deberá estar prevista una recuperación gradual de la velocidad conforme sea mayor el tiempo desde que se recibió la última notificación de congestión.

Como se ha explicado, los mensajes ICMP viajan encapsulados en el cuerpo de datos de los datagramas IP, por lo que no alteran su prioridad, es decir, se mezclan con el propio tráfico IP. Así, los datagramas encargados de avisar de la congestión circulan por la misma vía contribuyendo a aumentar la congestión. En sistemas al borde de la saturación, esto podría dar lugar a un crecimiento exponencial del tráfico al generarse mensajes de error para notificar los errores debidos a que los mensajes de error no hubieran podido ser entregados correctamente. Y si, a su vez, estos mensajes de error... En definitiva, la única forma de romper este círculo vicioso es hacer una excepción con los datagramas que transportan mensajes ICMP. Así pues, la especificación dice que los mensajes de error no pueden generar mensajes de error. Si un mensaje IP se descarta, simplemente se descarta, y no se envía notificación alguna de ello a nadie.

- *Problemas de fragmentación y prevención de bucles*

Al estudiar IP en el número anterior, se vio cómo un datagrama iba dotado de un campo TTL (*Time To Leave*) o de tiempo de vida máximo permitido. El valor de este campo se decrementaba en una unidad en cada nodo intermedio que atravesaba y, si llegaba a valer cero, el datagrama debía ser descartado. Esto se hacía en prevención de posibles bucles en la red. En caso de producirse la eliminación de un datagrama de la red debido a este motivo, el nodo que lo descarta envía a su remitente un mensaje ICMP *Time Exceded* (Tiempo Excedido por un Datagrama). Pero ésta no es la única situación capaz de generar este mensaje, existe otro límite temporal que también provoca



que un datagrama quede descartado. Para aquellos lectores que no "caigan" en ello, se recordará del mes anterior que un datagrama podía sufrir múltiples fragmentaciones en su tránsito por la red. Una vez en destino, se debía producir el proceso inverso de "desfragmentación", en el que se recuperaba el datagrama original en función de los fragmentos. Los fragmentos, que viajaban en forma de datagramas, podían a su vez llegar en desorden o perderse, por lo que el nodo destino activaba un temporizador al recibir el primero de estos. En caso de que este temporizador llegue a cero, se producirá otro mensaje *Time Exceded*. Para distinguir cual ha sido la causa de la generación de este mensaje, se asigna un código 0 a la finalización del TTL y un código 1 al time-out de desfragmentación.

- *Sincronización de relojes y estimación de tiempos de tránsito.*

En Internet, todas las máquinas funcionan con un reloj independiente lo que, en casos de desajuste considerable, puede dar lugar a comportamientos "extraños" de aplicaciones distribuidas o a efectos indeseados en sistemas de ficheros compartidos. Por ello, ICMP facilita también un mecanismo para la sincronización de relojes. Este mecanismo no es en absoluto riguroso ni preciso para aplicaciones de tiempo real o sistemas distribuidos, pero sí proporciona una aproximación aceptable si lo que se desea es que todas las máquinas de una red dispongan de la misma (más o menos) hora. Para ello, se definen otros dos mensajes ICMP: *Timestamp Request* y *Timestamp Reply* (solicitud y respuesta de marca de tiempos, respectivamente). De esta forma, si la máquina A quiere sincronizar su reloj con la máquina B, envía un *Timestamp Request* en el que graba la hora a la que ha sido enviado, A1. La máquina B, al recibirlo, le añade inmediatamente la hora de recepción B1 y lo reenvía al origen añadiendole también la hora de reenvío, B2. En el momento, A2, en que el datagrama vuelve al remitente, se dispone ya de suficientes datos temporales para realizar una estimación :

Tiempo de Tránsito (A -> B -> A) : Tt=A2-A1

Tiempo de Proceso en B : Tp=B2-B1

Se desea poner en hora A, por lo que los valores de horas de A no son fiables, pero sí servirán para calcular intervalos temporales. El tiempo invertido por el datagrama en circular por la red será el tiempo de tránsito total (Tt) menos el que ha perdido en el nodo B (Tt) es decir : Tt-Tp. Si admitimos que el tiempo de ida del datagrama es igual al de vuelta (aunque no sea cierto y puedan seguir rutas diferentes), el tiempo de uno cualquiera de estos dos recorridos será (Tt-Tp)/2. Así, para realizar una sincronización de relojes bastará asignar a A la hora que tenía B al reenviar el datagrama, más el tiempo transcurrido desde entonces, es decir, el tiempo de tránsito que se acaba de estimar para uno de los viajes:

Hora(A)=B2+(Tt-Tp)/2=B2+(A2-A1-B2+B1)/2

El lector comprobará que esta estimación es más bien un acto de fe, y hay que ser un verdadero creyente si se desea sincronizar con una máquina que se encuentre al otro lado del globo, pero el sistema funciona bastante bien en redes locales. La estimación de tiempos de tránsito es también muy útil para calcular las métricas de las distintas rutas, aunque para ello es necesario recurrir a métodos estadísticos sobre un conjunto mediciones continuas.

- *Determinación de máscaras de subred*

Otra de las funcionalidades que aporta ICMP es la posibilidad de preguntar a un gateway la máscara de subred empleada en la red local. Para ello se emplean otros dos mensajes : ICMP *Address Mask Request* e ICMP *Address Mask Reply*.

- *Otras situaciones*

Por último, ICMP notifica también otras situaciones especiales no contempladas hasta el momento, a las que asigna un código 12 y que suelen denominarse de "Parámetro Incorrecto" debido a que suelen estar ocasionadas por la especificación de opciones no soportadas o errores en campos de opción. También es necesario señalar que existen otros códigos que no se han tratado por haber quedado obsoletos.

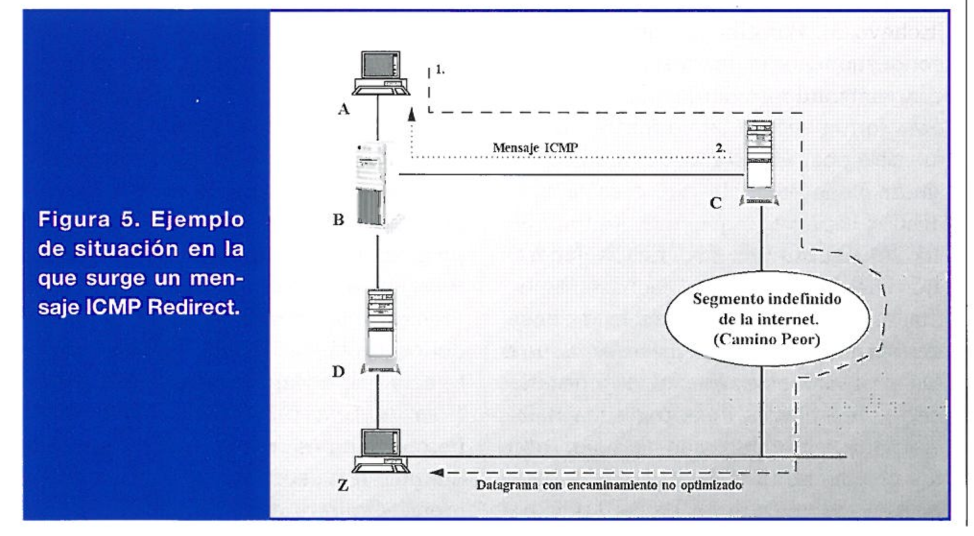


Figura 5. Ejemplo de situación en la que surge un mensaje ICMP Redirect.

CONTACTAR CON EL AUTOR

Para cualquier duda, comentario, sugerencia o crítica, se anima al lector a que se ponga en contacto con el autor a través de carta a la redacción de la revista o, preferiblemente mediante medios electrónicos con :

E-mail Internet : echeva@dit.upm.es

E-mail CompuServe : 100646,2456

WWW : http://highland.dit.upm.es:8000

[Echeva-1], "REDES TCP/IP: CONCEPTOS Y HERRAMIENTAS UNIX", núm. 23

[Echeva-2], "REDES TCP/IP: ARQUITECTURA Y PROTOCOLOS", núm. 25

[Echeva-3], "CHEQUEO A INTERNET: EL PROTOCOLO IP", núm. 26

La cita (*) pertenece a "Hamlet, Príncipe de Dinamarca" de William Shakespeare, acto I, escena IV.



EXCEPCIONES EN DELPHI

José M. González Ondina y Celestino Güemes Seoane

Al igual que otros conocidos lenguajes como el C++ o el ADA, el Object Pascal de Borland (el compilador incluido en Delphi) incorpora el manejo de excepciones. Esto permite crear programas menos sensibles a errores en tiempo de ejecución de forma sencilla y segura.

El manejo de excepciones, al igual que otras muchas características del Object Pascal de Delphi, no está definido en el Pascal estándar. Por lo tanto, antes de profundizar en el tema se hace necesaria una pequeña introducción a algunos conceptos.

¿QUÉ SON LAS EXCEPCIONES?

Las excepciones no son otra cosa que errores en tiempo de ejecución. Ejemplos conocidos de estos errores se producen cuando se intenta realizar una división por cero, escribir en un fichero que no ha sido abierto, acceder a posiciones de memoria protegidas, etc... Como se verá a continuación, las excepciones proporcionan un marco adecuado para tratar estos errores.

Además de las excepciones estándar definidas en Object Pascal, el programador puede definir las suyas propias para manejar nuevas situaciones de error en su programa. El acto de producir una excepción para tratar un error se denomina elevar (*raise*) la excepción, y el realizar acciones en respuesta a ésta manejarlas (*handle*).

Cuando un programa en Delphi incluye la unidad *SysUtils,* los errores en tiempo de ejecución son convertidos automáticamente en excepciones. De esta forma -como se verá más adelante- será posible tratarlos de manera elegante y segura.

EL MODELO DE EXCEPCIONES DE DELPHI

Las excepciones en Delphi están basadas en clases. Esto permite estructurar las excepciones en jerarquías y tratarlas con varios grados de aproximación. Por ejemplo, se podrían manejar igual todos los errores de tipo matemático o hacer diferencias entre estos.

Además, el tratamiento de excepciones está estructurado en bloques de código que pueden estar anidados unos dentro de otros en la misma forma que los bloques *begin..end*, lo cual da la posibilidad de hacer un seguimiento de éstas más o menos localizado.

Esta doble estructura jerárquica de las excepciones (por tipo de excepción y por bloques de código) permite una gran flexibilidad, pues se podría comenzar con un único manejador de excepciones que englobe todo el programa y todos los tipos de excepciones y posteriormente ir refinándolo, realizando acciones diferentes dependiendo del tipo de excepción y de su posición en el código.

UN PRIMER EJEMPLO

Aunque Delphi es un entorno esencialmente gráfico, en atención a los miles de programas escritos para funcionar en terminales en modo texto Borland ha incluido una unidad de compatibilidad llamada *WinCrt* cuya inclusión en la cláusula *Uses* permite compilar programas que utilicen los consabidos *Write* y *Read*. Esta característica es ideal a la hora de proponer ejemplos ya que elimina la necesidad de explicaciones como: "...Escoja ahora un botón de la barra de componentes e insértelo en el *form...*" tan comunes últimamente.

Para ver el manejo de excepciones en funcionamiento nada mejor que observar la diferencia en la ejucución de un pequeño programa con y sin ellas. Para ello se creará un proyecto del tipo *Crt Application* y se tecleará el listado del cuadro 1.

Hay que asegurarse de que la opción *Break on exception* que se halla en la pestaña de preferencias del diálogo que aparece al seleccionar *Enviroment* en el menú *Options* está desactivada. Esta

opción, cuando está activada, hace que un programa ejecutado desde el propio entorno de Delphi se detenga al elevarse una excepción, aunque exista un manejador para ella. Este comportamiento puede ser útil en algunos momentos de la depuración, pero puede ser molesto a la hora de probar los ejemplos.

Este programa compilará perfectamente, pero existen dos situaciones que podrían producir error en tiempo de ejecución. Una de ellas ocurre cuando se introduce un cero como valor de A (y B es distinto de cero), en ese caso se realiza una división por cero en el segundo Writeln del procedimiento *ImprimeSolucionesReales*. La otra situación de error ocurrirá cuando la variable *Discriminante* tenga un valor negativo, pues la función *Sqrt* sólo acepta valores positivos. En el primer caso, al realizarse la división se elevará una excepción *EZeroDivide* y en el segundo *EInvalidOp*.

UN MANEJADOR GLOBAL

Como se ha dicho antes, las excepciones se pueden tratar con mayor o menor detalle en cuanto al tipo de éstas. Esto es posible porque cuando se eleva una excepción, la información sobre ésta va guardada en una instancia de una clase derivada de *Exception*. Cada tipo de excepción tiene asociada una clase derivada diferente, y es la propia jerarquía de estas clases la que determina la estructura de los tipos de excepción.

Como ejemplo de esta estructura, a continuación se pueden ver algunas clases de excepciones definidas en la unidad *SysUtils*. Posteriormente se verá también cómo definir y elevar nuevas clases de excepciones.

```
EMathError=Class(Exception);
EZeroDivide=Class(EMathError);
EInvalidOp=Class(EMathError);
EOverflow=Class(EMathError);
EUnderflow=Class(EMathError);
```

Por supuesto hay muchas otras, las cuales pueden encontrarse en la ayuda del Delphi o en los manuales, pero estas bastan para este ejemplo. Lo primero que puede llamar la atención es que las nuevas excepciones derivan de las clases base sin añadir nada más. Esto, que en principio puede parecer bastante inútil, es sin embargo muy común. Generalmente, las nuevas excepciones no incluyen más información que la implícita que proporciona su posición en la jerarquía de clases. En este caso, la elevación de una excepción *EZeroDivide*, la cual se produce al realizar una división real por cero, puede tratarse como tal excepción *EZeroDivide*, pero también como una más general excepción *EMathError*, que engloba todas las excepciones matemáticas de números reales, o incluso como una *Exception*, la madre de todas las excepciones.

Como prueba de lo que ocurre en el programa anterior si no se manejan las excepciones, introduzca los grupos de valores 0,2,2 y 4,1,3. Con el primer grupo, el operador división eleva la excepción *EZeroDivide*, con el segundo, la función *Sqrt* eleva *EInvalidOp*. Ambas excepciones detienen el programa. Esta es una regla general, toda excepción que se produce y no es manejada detiene el programa, lo cual en el mejor de los casos produce una mala impresión.

Como primer ejemplo, se construirá un manejador global para atrapar todas las excepciones matemáticas que se produzcan en el programa. Para ello, se inserta todo el programa principal dentro de un bloque *try..except..end*, con lo cual cualquier excepción que se produzca entre ambas, aunque sea en bloques de código más internos (como el cuerpo de una función a la que se llame), transferirá el control al manejador de excepciones, es decir el código tras el *except*.

ESTRUCTURA DE UN MANEJADOR

En el manejador de excepciones va el código que ha de ejecutarse al producirse una excepción. Este puede responder por igual a cualquier excepción (para lo cual basta poner sin más las instrucciones que han de ejecutarse entre el *except* y el *end*) o actuar de forma diferente dependiendo del tipo de excepción. En este último caso la estructura del manejador será del tipo:

```
MANEJADOR:
on [identificador :] clase de excepción do sentencia;
on [identificador :] clase de excepción do sentencia;
  ...
[else lista de sentencias]
```

En el ejemplo considerado, el bloque principal del programa podría ser algo así:

```
begin
  try
      Writeln('Escribe los coeficientes A,B,C:');
    Readln(A, B, C);
    ImprimeSolucionesReales(A, B, C);
  except
    on EMathError do Writeln('Error matemático');
  end;
end.
```

De esta forma, al ocurrir una excepción en cualquier punto del programa, la ejecución se transfiere al manejador global. Si la excepción es de tipo matemático, es decir *EInvalidOp*, *EZeroDivide*, *EOverFlow* o *EUnderFlow*, se imprimirá la frase 'Error matemático'. Cualquier otra posible excepción que no sea de tipo matemático, "pasará de largo" el manejador (un ejemplo de esto ocurre si se introduce por ejemplo un carácter en lugar de un número para un coeficiente, en ese caso *Readln* eleva una excepción *EInOutError*). Una vez se ha ejecutado el código del manejador, el flujo del programa no retorna al lugar donde se produjo la excepción, sino que se sale del bloque *try..except..end* que se esté considerando y se ejecuta la siguiente instrucción tras el *end* del manejador. En el caso de que el manejador no trate la excepción producida la ejecución salta al manejador inmediatamente superior (recuérdese que los bloques de tratamiento de excepciones se pueden anidar) o se aborta el programa si es que no hay tal bloque superior.

Puesto que una cláusula *on..do* se ejecuta cuando la clase de la excepción actual coincide con la clase de excepción tras el *on* o es una clase derivada de ésta, ¿Qué ocurre cuando se produce una excepción que podría ser compatible con varias cláusulas *on..do* diferentes? En ese caso, Object Pascal ase-

gura que se aceptará la primera acción compatible que encuentre. En el ejemplo anterior, si se quisiera proporcionar una acción diferente para una división real por cero que para el resto de los errores matemáticos se podría poner:

```
except
  on EZeroDivide do Writeln('División por cero');
  on EMathError do Writeln('Error matemático');
end;
```

Si se quiere proveer al manejador de un comportamiento por defecto para cualquier excepción que no esté en ninguna cláusula *on..do* se utiliza la palabra clave *else*, la cual va al final del manejador, las sentencias tras el *else* se ejecutarán si no hay prevista una acción específica para la excepción. El manejador global podría ampliarse del siguiente modo:

```
...
except
  on EZeroDivide do Writeln('División por cero');
  on EMathError do Writeln('Error matemático');
else
  Writeln('Otro tipo de error');
end;
...
```

Si durante la ejecución de un manejador se produce una excepción, la que se estaba tratando es destruida y se eleva la nueva, que será tratada en el bloque de tratamiento de excepciones superior que la maneje o se terminará el programa si no existe tal bloque, pero nunca será manejada en el bloque actual.

Evidentemente, el ejemplo propuesto es muy sencillo y en realidad muy poco útil. El programa acaba informando que se ha producido un error, pero no hace nada por solucionarlo (podría por ejemplo pedir que se repitiesen los coeficientes o al menos especificar mejor el tipo de error), Además, al atrapar todas las excepciones que se producen en el programa no se puede conocer en qué punto se han producido y de esta forma, en la mayoría de los casos, no se podrá tomar la acción adecuada. Por último, sería conveniente

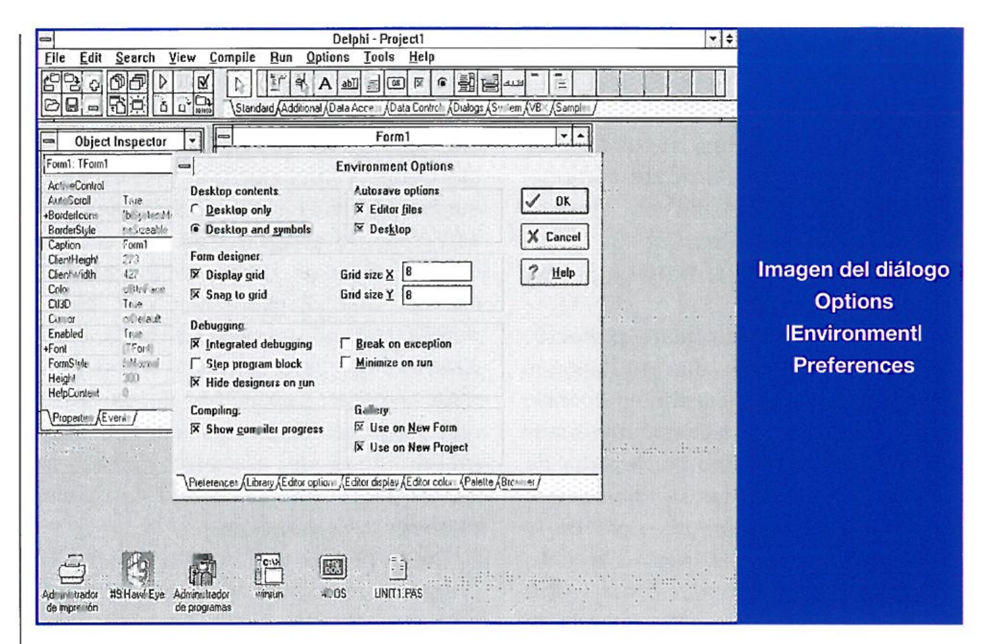


Imagen del diálogo Options |Environment| Preferences

que cuando la ecuación no tuviese soluciones reales, el procedimiento *ImprimeSolucionesReales* elevase una excepción de un nuevo tipo que expresase este hecho. Todos estos perfeccionamientos se verán a continuación.

NUEVOS TIPOS DE EXCEPCIÓN

Las excepciones proveen un medio estándar para la notificación de errores. Cuando no existe tal medio, cada cual se busca una forma propia para indicar situaciones excepcionales. Piénsese por ejemplo en la función *fopen()* de la librería del lenguaje C. Esta función abre un fichero, y devuelve un puntero al fichero si se ha podido abrir o el puntero nulo si ha habido algún problema. Así, cada vez que se llama a fopen() hay que comprobar el valor devuelto. Otras funciones notifican los errores de otras formas diferentes.

Las excepciones predefinidas de Object Pascal notifican errores que pueden producirse en los elementos predefinidos del lenguaje, pero ¿qué ocurre cuando se desea informar de un error que no puede incluirse en ninguno de esos tipos? En ese caso lo adecuado es definir nuevas clases de excepción.

Estas nuevas clases pueden derivarse de la clase base *Exception* o de cualquier otra clase derivada de ésta (en realidad no tienen por qué ser derivadas de *Exception*, pero no es buena idea hacerlo así). Como ejemplo se hará que el procedimiento *ImprimeSoluciones Reales* eleve una excepción *ECoeficientesInvalidos* (la cual se derivará de *EMathError*) cuando los valores de A, B y C no definan una ecuación con soluciones reales. La declaración del nuevo tipo de excepción podría ser:

```
ECoeficientesInvalidos=class(EMathError);
```

Esta declaración debe de ir en la sección *type* del programa o la unidad correspondiente.

Para elevar una excepción se utiliza la palabra clave *raise* seguida de una instancia del tipo de excepción que se quiere elevar. Puesto que se trata de una instancia, hay que crear un objeto de la clase de la excepción. Esto generalmente se realiza "sobre la marcha", escribiendo a continuación del *raise* una llamada a alguno de los constructores de la clase, en lugar de declarar una variable del tipo de la excepción, asignarle el resultado de la llamada a un constructor y hacer el *raise* con esa variable. Hay que tener en cuenta que los objetos excepción elevados son destruidos automáticamente tras ejecutar el código del manejador correspondiente. Por lo tanto, no hay que tratar de destruirlos en el cuerpo del manejador, ni intentar acceder a ellos al salir de éste en el caso de que previamente los hubiésemos asignado a una variable. El cuerpo de la función *ImprimeSoluciones Reales* será ahora:

```
begin
  Discriminante:=B*B-4*A*C;
  try
    Writeln('Solución 1: ', (-B-sqrt(Discriminante))/(2*A));
    Writeln('Solución 2: ', (-B+sqrt(Discriminante))/(2*A));
  except
    on EInvalidOp do raise ECoeficientesInvalidos.Create('Sin soluciones reales.');
    on EZeroDivide do raise ECoeficientesInvalidos.Create('No es de segundo grado.');
    { Cuando A es cero, no se trata de una ecuación de segundo grado. }
  end;
end;
```

Donde el constructor utilizado (puede encontrarse una descripción de todos ellos en la ayuda) es el más sencillo de todos. Toma una cadena como parámetro y la asigna al campo *Message* definido en *Exception*.

Por supuesto, se podrían haber puesto condicionales para comprobar si el discriminante era negativo o el coeficiente A igual a cero. Eso entra dentro de las preferencias de cada uno. En defensa de la solución expuesta aquí se puede decir que es más fácilmente extensible. Sin embargo, hay ocasiones en las que puede no ser conveniente dejar que se produzcan las excepciones (en el ejemplo propuesto, el procedimiento imprime algunas cosas aunque se produzca una excepción, lo cual queda bastante feo). Otra forma de realizar lo mismo sin dejar que se produzcan las excepciones puede ser:

```
begin
  Discriminante:=B*B-4*A*C;
  if (Discriminante<0.0) then
    raise ECoeficientesInvalidos.Create('Sin soluciones reales.');
  if (A=0) then raise ECoeficientesInvalidos.Create('No es de segundo grado.');
  Writeln('Solución 1: ', (-B-sqrt(Discriminante))/(2*A));
  Writeln('Solución 2: ', (-B+sqrt(Discriminante))/(2*A));
end;
```

```
program Ejemplo;

uses WinCrt, SysUtils;

var
  A, B, C: Real;

procedure ImprimeSolucionesReales(A, B, C: Real);
    { Este procedimiento halla y escribe en pantalla las dos posibles
      soluciones reales de la ecuación de segundo grado:  Ax^2+Bx+C=0. La
      solución se encuentra aplicando las fórmulas:
      solución1 = (-B-sqrt(B^2-4AC))/(2A)
      solución2 = (-B+sqrt(B^2-4AC))/(2A) }
var
  Discriminante: Real;

begin
  Discriminante:=B*B-4*A*C;
  Writeln('Solución 1: ', (-B-sqrt(Discriminante))/(2*A));
  Writeln('Solución 2: ', (-B+sqrt(Discriminante))/(2*A));
end;

{ PROGRAMA PRINCIPAL }
begin
  Writeln('Escribe los coeficientes A,B,C:');
  Readln(A, B, C);
  ImprimeSolucionesReales(A, B, C);
end.
```

CERRANDO EL CERCO

El siguiente paso es poner un bloque de tratamiento de excepciones más interno en el bloque principal del programa, para conocer con mayor precisión en qué punto del programa ha ocurrido el error. No hay que olvidar que en un programa un mismo tipo de excepción puede producirse un diversos sitios, y dependiendo de dónde ocurra tener un significado diferente y por lo tanto haber necesidad de tomar una acción diferente en respuesta a cada una de ellas.

En el ejemplo considerado se habían visto dos situaciones que elevaban una excepción, una cuando los coeficientes A, B, C producían algún error matemático (que ahora producirá una excepción *ECoeficientesInvalidos*) y otra cuando se introducía algo que no fuesen números como coeficientes. Cómo respuesta a ambas excepciones se tomarán ahora dos acciones ligeramente diferentes. En el primer caso se imprimirá en pantalla el mensaje previamente guardado en el campo *Message* del objeto excepción y se volverá a solicitar la introducción de los coeficientes. En el segundo caso, únicamente se repetirá la petición de coeficientes. El código para esto podría ser algo así:

```
repeat
    Writeln('Escribe los coeficientes A,B,C:');
  try
  Readln(A, B, C);
  ImprimeSolucionesReales(A, B, C);
  Repetir:=False;
  except
    on E:ECoeficientesInvalidos do
      begin
        Writeln(E.Message);
        Repetir:=True;
      end;
    on EInOutError do Repetir:=True;
  end;
until not repetir;
```

La variable Booleana *Repetir* se utiliza para indicar cuándo hay que repetir la introducción de datos. Y la construcción *on E:ECoeficientesInvalidos do ...* significa que el objeto excepción actual es accesible al manejador como la variable E de tipo *ECoeficientes Invalidos*.

BLOQUES DE PROTECCIÓN DE RECURSOS

Un uso típico de las excepciones es la creación de lo que se llama bloques de protección de recursos. El problema surge cuando un programa hace uso de alguna característica (recurso) que deba ser liberada una vez usada. Ejemplos de estos recursos son los ficheros (pues el numero de ellos que pueden estar abiertos a la vez es limitado) o la memoria dinámica. Si se produce una excepción cuando se ha obtenido uno de estos recursos y el manejador no lo libera, habrá un recurso "gastado", pero probablemente el programa se habrá olvidado de él y no podrá ser reutilizado por el sistema. Para resolver éste tipo de problemas Object Pascal proporciona la estructura *try..finally..end*.

La estructura es similar a la anteriormente vista *try..except..end*. Sin embargo existen dos diferencias, la primera es que la sección *finally* se ejecuta siempre, ya sea al producirse una excepción o al acabar la sección *try*. La segunda diferencia es que en la sección *finally* no hay un manejador de excepción, sólo hay código que se ejecutará sea cual sea el tipo de excepción producida.

Como ejemplo se verá un programa que abre un fichero, realiza ciertas accio-

nes con éste y luego lo cierra. Si en el transcurso de tales acciones se produce una excepción y el manejador de tal excepción se encuentra más allá de la instrucción que cierra el fichero, el fichero quedará abierto. Una forma de evitar esto sería:

```
AssignFile(fichero, 'stock.dat');
Reset(fichero);
try
  { En esta parte se realizan las acciones necesarias. }
finally
{ Esta parte se ejecuta en cualquier caso. }
CloseFile(fichero); { Libera el recurso. }
end;
```

Como puede verse, la obtención del recurso (en este caso abrir el fichero) está situada fuera del bloque *try..finally*, que es lo que se podría llamar parte protegida. Cualquier excepción que ocurra dentro de este bloque obligará a ejecutarse el código tras el *finally*, que es donde se deben poner las instrucciones que liberan los recursos. Si no se produce ninguna excepción, el flujo del programa pasará a la sección *finally* igualmente al terminar la parte protegida, liberando de esta forma los recursos en cualquiera de los dos casos.

Algo importante a tener en cuenta es que el código de terminación (entre el *finally* y el *end*) no es un manejador de excepciones. Cuando se produce una excepción se ejecuta este código de terminación, pero a su fin, la excepción vuelve a ser elevada y ha de ser manejada en un bloque *try..except..end*. El bloque de protección de recursos sólo sirve para asegurarse de que el recurso es liberado siempre.

RELANZANDO EXCEPCIONES

A veces puede ocurrir que se desee tratar una excepción en un manejador local sin eliminar el tratamiento que de ella hace otro bloque superior o tratar una excepción sólo cuando se cumplan determinadas condiciones, dejando para el bloque superior los otros casos. Generalmente, cuando una excepción es atrapada en un bloque de tratamiento de excepciones, el objeto excepción es destruido de forma automática en el manejador, de forma que el bloque superior no es nunca llamado. Para evitar este comportamiento, se utiliza la sentencia *raise* sin argumento. Cuando se utiliza *raise* en esta forma la excepción actual es relanzada de forma que pueda ser atrapada en otro bloque superior que provea un manejador para ella. Puesto que sólo en los manejadores de excepción existe el concepto de excepción actual, sólo dentro de éstos puede ponerse un *raise* sin argumento.

Un ejemplo de esto podría ser:

```
function Exponencial(Base, Exponente: real): real;
begin
 try
  Result:=Exp(Exponente*Ln(base));
 except
        on EInvalidOp do if (Frac(Exponente)=0) then
  begin
       if (Frac(Exponente/2)=0) then Result=Exp(Exponente*Ln(Abs(Base)))
else   Result=-Exp(Exponente*Ln(Abs(Base)));
  end else raise;
 end;
end;
```

La función del ejemplo devuelve el resultado de elevar *Base* a *Exponente*. Puesto que para ello calcula el logaritmo de la base, si ésta es un valor negativo se genera una excepción, ya que la función logaritmo sólo acepta números positivos. En ese caso, la función atrapa la excepción y comprueba si el exponente es entero. La función exponencial de una base negativa sí está definida si el exponente es entero (por ejemplo, -2 elevado a 3 es -8), siendo el resultado negativo si el exponente es impar y positivo si es par. Si el exponente no es entero, no se puede calcular el resultado y se relanza la excepción de forma que pueda ser atrapada por otro manejador por encima del actual.

EL EVENTO ONEXCEPTION

Como se ha visto antes, para poner un manejador global puede ponerse un bloque de tratamiento de excepciones justo entre el *begin* y el *end* que definen el programa principal, pero ¿cómo poner un manejador global en una aplicación que no sea del tipo *Crt*?. En las aplicaciones basadas en *forms* no existe un programa principal, sólo código que responde a eventos. Para poder hacer esto, el objeto *Application* que define una aplicación dispone de un evento llamado *OnException*, el cual acepta procedimientos (en realidad métodos, generalmente pertenecientes al *form*) de la forma:

```
procedure Nombre(Sender: TObject; E: Exception);
```

Donde *Sender* identifica el componente en el que se ha producido la excepción y E el tipo de excepción producida. Puesto que el objeto *Application* no es accesible desde el *Object inspector*, la asignación de un procedimiento a este evento ha de hacerse por programa. El sitio más adecuado para hacer esto es el evento *OnCreate* del *form* que es llamado en el momento en que éste es creado.

En el listado siguiente puede verse la estructura general de un programa que hace uso de esta característica. El programa responde de forma diferente a las excepciones *EZeroDivide*, *EMathError* y *Exception*, de forma similar a como podría haberse hecho en un manejador con estructuras *on..do*.

```
procedure TForm1.FormCreate(Sender: TObject);
begin

Application.OnException:=HandleGlobal
;
end;

procedure          TForm1.HandleGlobal
(Sender: TObject; E: Exception);
begin
 if (E is EZeroDivide) then ...
 else if (E is EMathError) then ...
 else if (E is Exception) then ...;
end;
```

No hay que olvidar, por supuesto, incluir la declaración del método *TForm1.HandleGlobal* en la parte pública de *Tform1*.

El funcionamiento de este pseudo-manejador (pues no hay que olvidar que en realidad no lo es) es sencillo. En las sentencias *if* se comprueba el tipo de la excepción que ha provocado el evento *OnException*, y el uso de la estructura *if..else if..else if...* es necesario para evitar que una misma excepción tenga varias respuestas.

SISTEMAS OPERATIVOS DE RED (II)

María Jesús Recio

En el artículo anterior se definieron los tipos de sistemas operativos capaces de gestionar redes de área local, así como algunas filosofías de implementación para dichos sistemas.

Volviendo atrás, se debe mencionar que existen sistemas diseñados exclusivamente para el manejo de la red, y que por tanto incorporan herramientas muy potentes para la gestión de elementos como son el tráfico de la red, la configuración y manejo de diferentes protocolos de comunicaciones, usuarios, dispositivos hardware como impresoras, etc... frente a otros sistemas que lo que hacen es incorporar herramientas básicas de comunicaciones que permiten el intercambio de información en la red, incorporando protocolos de comunicaciones que además se pueden encontrar de manera independiente.

Se habló de dos filosofías básicas con las que se implementan los distintos sistemas operativos de red: orientada a servidor y de igual a igual, además de comentar una arquitectura muy difundida en la actualidad, que es la arquitectura Cliente-Servidor.

Como sistema operativo de red propiamente dicho, el que más auge está teniendo en la actualidad en el mercado es Netware de Novell. Se trata de un sistema que en su última versión (4.1) es capaz de gestionar no sólo redes de área local, sino también redes de área extensa.

Como ejemplos de sistemas operativos que lo que hacen es incorporar herramientas para las comunicaciones se puede citar Windows 95.



FUNCIONES DEL SOFTWARE DE RED

En los sistemas operativos basados en la arquitectura Cliente-Servidor, los elementos y las funciones de la máquina servidor difieren en gran medida de lo que debe instalarse en las máquinas clientes, aunque obviamente existe una parte común.

MÁQUINAS SERVIDOR

En la máquina servidor se tendrá, por un lado, el software de comunicaciones, que será la parte del sistema que permita la comunicación a través de la red de área local con las diferentes máquinas que la componen, y por otro lado, el sistema operativo de red, que incorporará, además de las tareas típicas de un sistema operativo, tareas propias de un sistema de red, como son manejo de dispositivos, suministro de recursos, creación y gestión de procesos, control de acceso y autentificación de usuarios.

Las funciones de los servidores dependerán en gran medida de la naturaleza del servicio que ofrezcan

Las funciones de los servidores dependerán en gran medida, de la naturaleza del servicio que ofrezcan, es decir, no realizará las mismas tareas un servidor de ficheros que uno de impresión. En

En la actualidad, la mayoría de los sistemas operativos de red incorporan herramientas para la gestión de red, eso sin mencionar el número elevado de sistemas diseñados exclusivamente para el manejo de una red y gestión de la misma, así como los protocolos de comunicaciones que permiten la conexión entre diferentes ordenadores.

los siguientes puntos se definirán las características dependiendo del tipo de servidor.

- Servidor de ficheros.

Se puede dividir el software del sistema de los servidores de ficheros en tres partes:

- El sistema de gestión de ficheros, que escribe y lee los datos en una o más unidades de disco.
- El sistema de caché de disco, que mantiene los datos entrantes y salientes en memoria RAM, lo que permite un tratamiento más rápido de lo que ofrecen las posibilidades físicas de un disco duro.
- El sistema de control de accesos, que controla quién puede utilizar los datos y de qué forma acceden a los ficheros varias aplicaciones simultáneamente.

Estos elementos de software del servidor de ficheros son aplicaciones realmente sofisticadas, que se ejecutan sobre un sistema de gestión de ficheros. En algunos productos, como puede ser LANtastic, esta parte se basa en la de MS-DOS como sistema de gestión de ficheros (redes de igual a igual), pero en muchos otros productos como Netware de Novell y VINES de Banyan la aplicación y los sistemas de gestión de ficheros están altamente integrados. La función básica del sistema de gestión de ficheros es mover las cabezas de la unidad de disco duro y entregar los datos a las estaciones clientes por medio de la red, independientemente del sistema operativo que estas tengan.

Los algoritmos con los que se implementan las funciones están altamente optimizados para así disminuir el tiempo empleado en servir las peticiones. Debe pensarse que, mientras en un PC con DOS se realiza una única petición de disco en cada instante, a un servidor de ficheros de una red le pueden llegar muchas peticiones a la vez, por lo que debe incorporar herramientas muy potentes para poder servir en un tiempo satisfactorio dichas peticiones.

- Servidor de impresión.

Los servidores que actúan como servidores de impresión hacen que las impresoras estén disponibles en la red para su uso compartido. Se encargan de recibir peticiones de impresión que almacenan un unos directorios especiales para posteriormente imprimir.

Su software debe estar capacitado para dirigir las peticiones a las colas de impresión y de éstas a las impresoras. Asimismo deben tener algoritmos de ordenación y planificación de las colas. Suele definirse, al menos, una cola de impresión por impresora disponible. En esta cola se irán almacenando los trabajos pendientes de imprimir de acuerdo al algoritmo que la gestione: el primer trabajo de impresión en llegar será el primero en salir, o bien el trabajo más corto el próximo en imprimirse. Cuando un cliente envía un trabajo de impresión puede hacerlo, o bien para que se imprima en la primera impresora libre, o bien para que se imprima en una impresora determinada.

A veces, los propios servidores de ficheros son también servidores de impresión, por lo que un una misma máquina debe incorporarse el software necesario para ambos servidores.

- Servidores de comunicaciones.

El software implantado en un servidor de comunicaciones cubre muchas tareas, desde realizar la función de pasarela hacia otras máquinas hasta ofrecer servicios de acceso a un módem. Las principales funciones incorporadas implican la capacidad de establecer enlaces de los siguientes tipos:

- Entre redes de área local distintas.
- Entre red de área local y red de área extensa.
- Entre red de área local y mainframe.
- Servicios de fax.

Un factor muy importante a la hora de seleccionar una máquina como servidor de comunicaciones es la capacidad de su CPU, ya que este servidor debe proporcionar conexiones en tiempo real entre las estaciones clientes y los canales de comunicación.

MÁQUINA CLIENTE

En una máquina cliente el software se tiene que encargar de funciones diferentes que las vistas para las máquinas servidor. En este caso no se tiene que realizar búsquedas de ficheros, ni gestionar impresoras. Las tareas a realizar son mucho menos complejas.

El software de red para las máquinas cliente está dividido en tres categorías:

- Software redirector: Se encarga de capturar las peticiones realizadas desde la máquina cliente en la que está instalado, comprobar si esas peticiones son dirigidas a servicios que puede proporcionar el sistema operativo local y, si no es así, desviar dichas peticiones hacia el servidor apropiado de la red.
- Software de comunicaciones: Este módulo aparece tanto en las máquinas clientes como en las servidoras. Se trata del software encargado de garantizar que los mensajes se transmitan por la red de forma segura y libres de errores, dividir los mensajes en trozos más pequeños (paquetes) según las características de la red, garantizando que lleguen al destino correcto, y finalmente proporcionar a los programas de aplicación una forma de solicitar servicios a la red.
- Controlador de la tarjeta de interfaz de red: También este módulo es común en máquinas clientes y servidoras. Se encarga de enlazar el

La función básica del sistema de gestión de ficheros es mover las cabezas de la unidad de disco duro y entregar los datos a las estaciones clientes

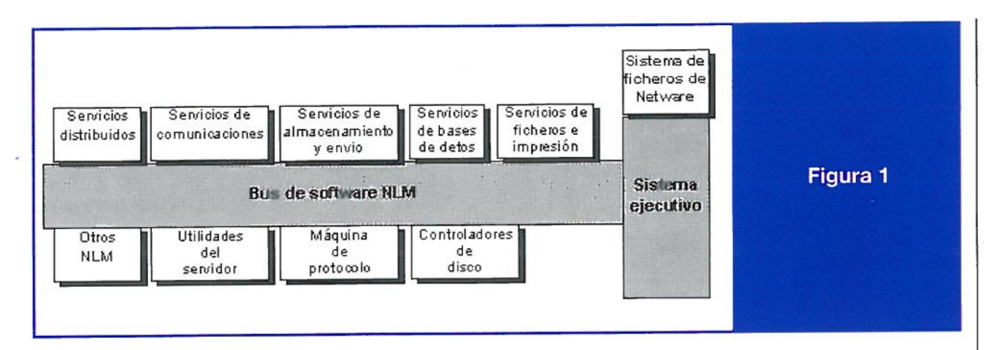


Figura 1

software de transporte con la tarjeta de red.

WINDOWS PARA TRABAJO EN GRUPO

Se trata de una aplicación gráfica que funciona sobre MS-DOS para redes de igual a igual. Incorpora un protocolo de comunicaciones que permite la interconexión de PC's utilizando el entorno gráfico Windows, y en el que cada ordenador aporta al resto sus propios recursos, como son ficheros e impresoras.

No incorpora herramientas para la gestión de la red, ni control de usuarios, sistema de seguridad, etc... y las funciones de conectividad están totalmente integradas en el entorno gráfico y de fácil manejo.

Las funciones de red en Windows para Trabajo en Grupo están distribuidas entre tres elementos del entorno: el Administrador de Ficheros, el Administrador de Impresión, y el Panel de Control.

- En el Administrador de Ficheros se hace posible el acceso a discos de otras máquinas de la red, es decir, a directorios y ficheros. Estas tareas se realizan a través de una nueva barra de herramientas. El acceso a ficheros y directorios se puede establecer a distintos niveles de compartición: sólo lectura, acceso libre o acceso con contraseña. La compartición de un directorio implica automáticamente la de los subdirectorios situados por debajo de él. No hay forma de proteger ficheros de manera individual.
- En el Administrador de Impresión se indica cuáles de las impresoras locales pueden ser compartidas, es decir, utilizadas por otras máquinas de la red.
- En el Panel de Control aparece una nueva utilidad, que permite la configuración del entorno de red: tarjetas, contraseñas, nombre de la máquina, protocolos de comunicación, etc. Además, desde aquí se puede modificar la distribución del tiempo que la máquina dedica a peticiones locales y a peticiones remotas.

Además, en Windows para Trabajo en Grupo se incorporan dos aplicaciones que facilitan el paso de mensajes entre usuarios: *Mail* (correo electrónico) y *Schedule+*, que se trata de una agenda compartida por los miembros de un grupo de trabajo.

WINDOWS 95

Se trata de un sistema operativo gráfico que incorpora funciones de red más sofisticadas que Windows para Trabajo en Grupo. Además de ofrecer gestión de recursos, incorpora diferentes familias de protocolos que facilitan su conexión a través de red con otros sistemas diferentes. Entre ellos se pueden destacar los protocolos TCP/IP, IPX, NetBeui, PC-NFS, DLC, etc.

Al igual que en Windows para Trabajo en Grupo, la forma de acceder a los recursos de la red se realiza a través de tres puntos: Panel de Control, Explorador de Ficheros (nueva versión del Administrador de Ficheros), y a través del sistema de impresión, sin contar las utilidades que incorpora para el correo electrónico, acceso remoto vía módem, *Schedule+*, etc.

LANTASTIC

Se trata de un sistema operativo de igual a igual, en el que cada ordenador de la red puede actual tanto como cliente como servidor, aunque también incluye la posibilidad de mejorar las prestaciones de compartición de ficheros, estableciendo servidores dedicados.

Aunque inicialmente se había desarrollado como un sistema operativo que únicamente permitiera la compartición de recursos entre los diferentes ordenadores de la red, en la actualidad incorpora herramientas complejas, como son:

- Seguridad. Ofrece tres niveles de seguridad: seguridad de *Login*, seguridad mediante lista de control de accesos y seguridad mediante anotaciones de seguimiento.
- Funciones de impresión sofisticada.
- Correo electrónico.
- Caché de disco.
- Compartición de CD-ROM.
- Capacidades de voz.
- Interfaz Windows.

LANtastic trabaja tanto en entorno DOS como en entorno Windows.

Para poder compartir un recurso es necesario darle un nombre. Esto se hace tanto para recursos software como hardware.

NETWARE

Netware de Novell es un sistema operativo para redes basadas en servidor que se implanta en redes de PC's que están funcionando con MS-DOS o Windows. Se trata de un sistema especializado en ofrecer servicios de red, fundamentalmente sevicios de ficheros, para lo que dispone de herramientas como disco espejo, disco

A veces, los propios servidores de ficheros son también servidores de impresión

duplicado, control de concurrencia, control de transacciones.

Los protocolos originales incorporados en este sistema operativo eran IPX/SPX, aunque en la actualidad integra otros protocolos, como son TCP/IP, *Apple Talk*, y protocolos OSI. Tanto el soporte para estas familias de protocolos como para los servicios de ficheros se hace de forma modular.

Se trata de un sistema en el que las máquinas clientes acceden a la máquina servidor de ficheros como si estuviesen accediendo a un disco duro local, es decir, simplemente indicando una letra que representa a dicho disco. Cuando se escribe una orden en una máquina cliente, el software redirector dirige la petición o bien al DOS de la propia máquina, o bien hacia un servidor de ficheros. Algunas características importantes de este sistema son:

- Sistema modular: Está compuesto por una serie de módulos cargables NLM (*Netware Loadable Modules*) que permiten modificar las funciones que el servidor realiza (ver figura 1).
- Gestión de espacio de disco duro en el servidor.
- Auditoría de sucesos en la red: Permite monitorizar sucesos relacionados con el sistema de ficheros, la gestión de colas, etc...
- Sistema de comunicaciones mejorado con un mecanismo de ventana que agiliza la transmisión de información, entre otras cosas.
- Gestión de memoria en el servidor: Tiene un esquema de gestión de memoria que se adapta a servidores que tienen que dar servicios a cientos de usuarios.
- Los Servicios de Directorio de Netware (NDS): Se trata de una base de datos que contiene toda la información sobre la totalidad de recursos de la red.

Windows 95 incorpora diferentes familias de protocolos que facilitan su conexión a través de red con otros sistema diferentes

Los NDS gestionan todos los recursos de la red como si se tratase de objetos, definiendo para ello dos tipos de objetos: objetos contenedores y objetos terminales. Los objetos contenedores son aquellos objetos que pueden contener otros objetos. Se trata de una forma de organización de los recursos. Los objetos terminales, por su parte, son objetos finales que no pueden contener otros objetos y que representan los recursos de la red. Existen diferentes objetos terminales como son: usuarios, grupos, impresoras, volúmenes, perfiles, alias, *bindery*, etc.

VINES

Es un sistema operativo para redes de área local basadas en servidor. Se basa en UNIX System V y dispone de diferentes módulos software para las máquinas clientes y para los servidores. Toda la organización de la red se fundamenta en una base de datos llamada *StreetTalk*, en la que se mantiene información actualizada referente a las características del sistema, usuarios, servicios disponibles y conexiones realizadas.

Soporta diferentes protocolos de comunicaciones: TCP/IP, *Apple Talk*, NetBIOS, protocolos orientados de Banyan, etc... Los principales servicios de red ofrecidos por este sistema son:

- Incorporación de una utilidad para correo electrónico.
- Permitir la utilización de impresoras compartidas en la red que pueden estar conectadas tanto en el servidor como en las máquinas clientes.
- Incorporación de una utilidad para comunicaciones interactivas entre distintos usuarios de la red.
- Permitir a los usuarios la utilización de ficheros de la red, así como la definición de modalidades de acceso a dichos ficheros.
- Incorporación de un servicio de nombres: se trata de una base de datos donde se representan todos los recursos de la red. Estos recursos se encuentran organizados en grupos y listas. Con este servicio es posible ver la red como un conjunto de recursos, sin tener en cuenta qué servidor, en concreto, proporciona dichos recursos.

LAN MANAGER

Se trata de un sistema de ficheros y de gestión de red que se ejecuta sobre el sistema operativo OS/2. Tanto es así que se puede decir que el gestor de red es sólo una tarea más que ejecuta el sistema OS/2 (recuerde que se trata de un sistema operativo multitarea). En la tabla 1 se muestra la arquitectura de OS/2-LAN Manager.

En la actualidad Microsoft está reemplazando este sistema operativo por Windows NT.

WINDOWS NT LAN MANAGER.

Se trata de un sistema operativo que ofrece grandes prestaciones de control y gestión de red, sin hablar de sus características como sistema operativo (multiprocesamiento simétrico, sistemas de seguridad, modo de operación de 32 bits).

Las facilidades de red incorporadas en Windows NT permiten la compartición de ficheros del sistema local con otros usuarios de la red y la conexión con directorios compartidos de otros sistemas. Es compatible con Windows para Trabajo en Grupo, y además ofrece mecanismos de conexión a sistemas diferentes como puede ser UNIX.

PRÓXIMO NÚMERO

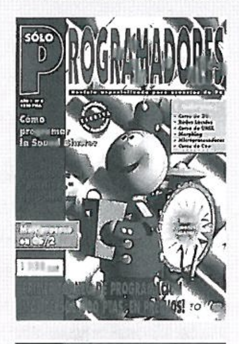
En el próximo número se hará un repaso por diferentes protocolos de comunicaciones para redes de área local. En el artículo 5 de esta serie de vieron en detalle los protocolos TCP/IP. En la próxima entrega se verán otros protocolos también muy difundidos, como son IPX, NETBIOS, etc.

CÓMO SUSCRIBIRSE A































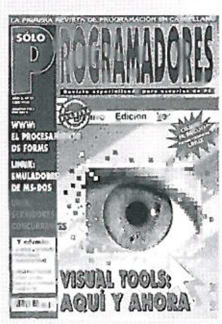












scríbase enviando este cupón por correo o fax (91) 661.43.86, o llamando al teléfono (91) 661.42.11 Horario 9 a 14 y 15:00 a 18:00 h.

)eseo suscribirme a la revista SÓLO PROGRAMADORES acogiéndome a la siguiente modalidad:

☐ Suscripción: 1 año (12 números) **por sólo** 11.950 ptas. (ahorro 20%). ☐ Estudiantes carreras técnicas: 8.950 ptas. (ahorro 40%)

' ESTA OFERTA ANULA LAS ANTERIORES, DESCUENTOS NO ACUMULABLES.

\lombre y apellidos...Domicilio...

'oblación..............................C.P ....Provincia..................Telf..........................Profesión..................

:ORMA DE PAGO:

☐ Con cargo a mi tarjeta VISA nº ____ ____ ____ ____

Fecha de caducidad de la tarjeta................Nombre del titular, si es distinto..............................

☐ Domiciliación bancaria.

Señor Director del banco ....

Población ....

Ruego a vd. que se sirva cargar en mi ☐ cuenta corriente ☐ libreta

CODIGO CUENTA CLIENTE

ENTIDAD	OFICINA	DC	Nº CUENTA

: ahorro número...

recibo que le será presentado por TOWER COMMUNICATIONS, S.R.L.

mo pago de mi suscripción a la revista SÓLO PROGRAMADORES.

Firma:

☐ Contra-reembolso del importe más gastos de envío.

☐ Cheque a nombre de TOWER COMMUNICATIONS S.R.L., que adjunto.

☐ Giro Postal (adjunto fotocopia del resguardo).

Rellena este cupón y envíalo a:
TOWER COMMUNICATIONS SRL
C/ Aragoneses, 7
28100 Pol. Ind. ALCOBENDAS (Madrid)

EL SOMBREADO GOURAUD

Pedro Antón Alonso

Al avanzar en el curso de 3D, se hace totalmente imprescindible crear distintos tipos de rellenados o sombreados. El sombreado plano es insuficiente para crear un efecto algo real. Esta nueva entrega trata el rellenado de triángulos con sombreado gouraud.

Se ha cambiado la figura del donut por la del pato, como se comentó vagamente en el artículo anterior. Destacar, una vez más, que el objetivo primordial de este curso no es conseguir rutinas de gran eficiencia en lo que a velocidad se trata, sino conseguir rutinas claras, con una relativa eficiencia. Existen, obviamente, partes del código de los fuentes que acompañan a este curso que pueden ser optimizadas o incluso muy optimizadas. Trabajo que puede resultar altamente atractivo para los lectores que siguen esta serie de entregas. Se anima, por tanto, desde estas líneas a todos los lectores a implementar las rutinas en otros lenguajes e intentar optimizarlas.

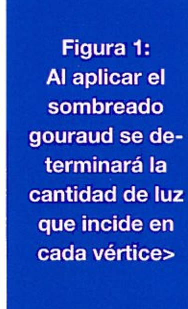
Al aplicar el sombreado gouraud se determinará la cantidad de luz que incide en cada vértice

LA TERCERA DIMENSIÓN Y LA LUZ

"Los objetos físicos son sólidos" era una frase con la comenzaba la anterior entrega. Eso es algo sobre lo que no cabe la menor duda, pero ninguna superficie plana va a recibir, normalmente, igual cantidad de luz en todos sus puntos.

De acuerdo con la "Ley de Lambert", la intensidad de luz que incide en un plano es directamente proporcional al ángulo existente entre el vector de luz incidente y el vector normal del plano. Para determinar dicho ángulo se puede usar el producto escalar de dos vectores, ya que este es igual al producto de los módulos de cada uno de ellos por el coseno del ángulo que forman. De esta forma, se puede calcular el ángulo que forman dos vectores según se muestra en la fórmula 1.

Para aplicar el sombreado gouraud, en lugar de calcular la cantidad de luz que incide en un plano se determinará la cantidad de luz que incide en cada vértice. En el ejemplo que acompaña a este artículo ésta depende directamente de la distancia z del punto al observador. Esto simula una luz justo delante de nuestra nariz, pero es algo que



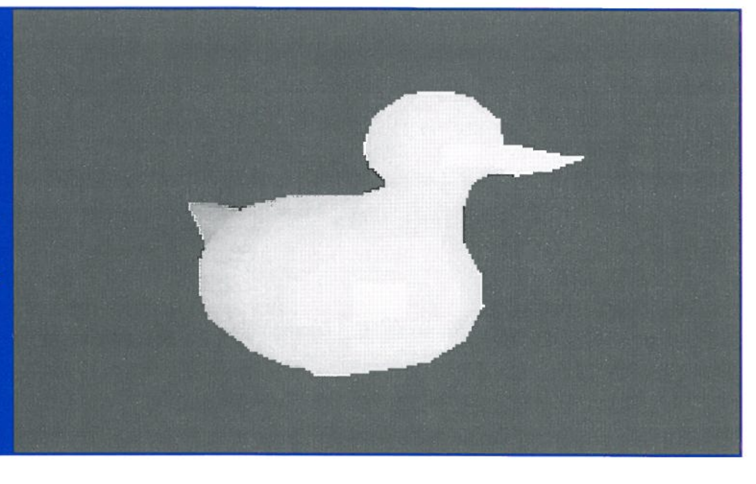


Figura 1: Al aplicar el sombreado gouraud se determinará la cantidad de luz que incide en cada vértice>

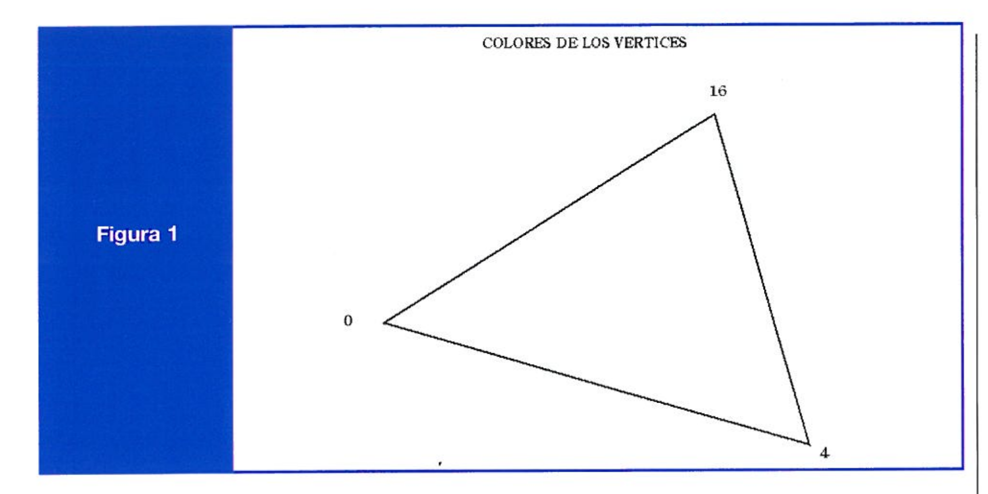


Figura 1

puede variarse para conseguir efectos asombrosos si se calcula la intensidad media de luz que recibe cada vértice calculando la media aritmética de la intensidad de luz que recibe cada plano que comparte el vértice el cuestión, algo que se hará en próximos artículos de este curso. De cualquier forma, cada vértice del triángulo tendrá además de unas coordenadas (x,y) un color.

EL SOMBREADO GOURAUD

El sombreado gouraud se realiza pintando líneas horizontales con un degradado de color.

Recordemos los pasos seguidos en el anterior artículo para calcular los incrementos de las aristas de un triángulo:

1- Definir dos listas de incrementos, una para la derecha y otra para la izquierda.
2- Calcular los incrementos de la arista más larga y almacenarla en la lista de la izquierda.
3- Convertir las otras dos aristas y almacenar los incrementos en la lista de la derecha.

Estos pasos serán idénticos en ambos rellenados. Ahora bien, en el rellenado Gouraud se debe tener en cuenta el color de cada vértice del triángulo, y con esos datos calcular el color de cada punto del triángulo usando interpolación lineal.

En primer lugar se interpolarán los colores de los vértices de las aristas, para después poder interpolar los colores de cada línea horizontal entre los colores de los extremos de la línea.

Estos pasos y su resultado final están claramente ilustrados en la Figura 1.

La interpolación de colores se realizará de forma análoga a la interpolación de los incrementos de las aristas. Por ello, en el rellenado Gouraud, además de interpolar los valores de x se interpolarán también los valores de color.

Esto hará el cálculo de los incrementos de las aristas un poco más complicado, ya que además de interpolar los valores de x se interpolará el color. Pero ahora se usará punto fijo 8.8 ya que los valores de color, en la resolución que se está empleando, nunca sobrepasarán 255.

Las tablas empleadas ahora para calcular los incrementos tendrán dimensión dos. De esta forma, en Tabla[coor_y,0] se almacenará el valor de la coordenada x de la arista para el valor de y coor_y en Tabla[coor_y,1] el valor de color correspondiente a dichas coordenadas (x,y).

Para rellenar el triángulo se dibujan líneas horizontales, al igual que ocurría con el rellenado sólido. Pero a cada línea horizontal, en este caso, le corresponden cuatro valores, un par de valores de x y color iniciales y otro par de x y color finales. Normalmente, los valores de color inicial y final no serán el mismo. Por lo tanto, también se necesitará interpolar los colores a largo de la línea que se van a dibujar. Interpolación que se calculará de igual forma que la de las aristas, con una sola diferencia. Ahora las variaciones de color serán con respecto de x en lugar de con respecto de y.

Es interesante destacar, al igual que ocurría con el rellenado plano, la necesidad de pintar las líneas horizontales de izquierda a derecha, teniendo en cuenta que si se han de intercambiar los valores de x también deben de intercambiarse los valores de color.

Las líneas horizontales con el degradado de color se dibujarán entonces conocidos cinco valores:

- coordenada x inicial.
- coordenada x final.
- color inicial.
- color final.
- coordenada y .

LISTADO 1

```
mov bx,XH1
 mov cx,XH2
 sub cx,bx      { Lon }
 jz @@NoLine
 mov ax,SegDes
 mov es,ax
 mov di,CntY
 mov dx,di
 xchg dh,dl
 shl di,6
 add di,dx
 add di,bx
 mov bx,CH1
 mov dx,IncColH
@@Another:
 mov es:[di],bh
 add bx,dx
 inc di
 dec cx
 jnz @@Another
@@NoLine:
```

El proceso a seguir es el siguiente:

1 Determinar la dirección de memoria donde se comenzará a dibujar la línea.
2 Calcular la longitud de la línea.
3 Calcular el incremento de color para cada pixel.
3 Escribir los bytes del color adecuado sumando a cada byte el incremento de color calculado en el paso 2.

Una solución bastante eficiente es la mostrada en el listado 1.

Se puede observar cómo se usa la parte alta del byte a pintar. Este pequeño truco, que ahorra una rotación en el bucle principal, es posible debido a que se está empleando punto fijo 8.8.

EL CLIPPING

Como ya se comentó al hablar del sombreado sólido, recortar las imágenes cuando éstas se salen de la pantalla es algo absolutamente necesario.

Recortar el polígono en y es bastante sencillo y se realizará de igual forma que con el rellenado plano. Bastará con incluir un par de condiciones a la rutina.

Pero ahora se ha de mostrar especial atención al recorte en x, ya que el color

Formula 1

$$\vec{a} = (a_x, a_y, a_z)$$

$$\vec{b} = (b_x, b_y, b_z)$$

$$\cos \alpha = \frac{a_x b_x + a_y b_y + a_z b_z}{\sqrt{a_x^2 + a_y^2 + a_z^2}\ \sqrt{b_x^2 + b_y^2 + b_z^2}}$$

también quedará recortado. Por lo tanto, se debe calcular el recorte de x y de color como se observa en el listado 2, donde se puede ver la rutina de rellenado Gouraud completa.

LA ILUMINACION

Una correcta iluminación hará que el sombreado Gouraud sea más vistoso.

Iluminando correctamente el objeto a tratar se consigue una mejor sensación de relieve. En el ejemplo incluido en el CD-ROM de la revista no se calcula correctamente la intensidad de luz que incide sobre cada vértice del triángulo. No obstante, en posteriores entregas se explicará adecuadamente la forma de hacerlo, algo totalmente imprescindible a la hora de simular un rellenado con sombreado phong.

En este artículo la iluminación se toma, como ya se ha dicho, dependiendo de la profundidad. Esto servirá para salir del paso, pero no simula correctamente una luz, y en consecuencia la intensidad de luz que incide sobre cada vértice no es del todo correcta.

LA UNIDAD DEMOVGA

Se ha incluido en la unidad la rutina de sombreados Gouraud con clipping. También se han modificado los valores de clipping de pantalla, siendo ahora variables globales de la librería. De esta forma se reduce considerablemente el número de parámetros que se le pasan a las rutinas. Los pasos que sigue la rutina de sombreados Gouraud son:

1- Hacer que el punto más alto del triángulo sea el primero.
2- Poner la arista más larga del triángulo entre los puntos primero y segundo.
3- Calcular los incrementos de x de las tres aristas, recortados en y, así como los incrementos de color.
4- Recortar los limites de y del triángulo si es necesario.
5- Pintar todas las líneas horizontales con el degradado de color, dentro de los limites de y, comprobando si la línea a pintar es de izquierda a derecha o de derecha a izquierda, así como el incremento de color de la línea.

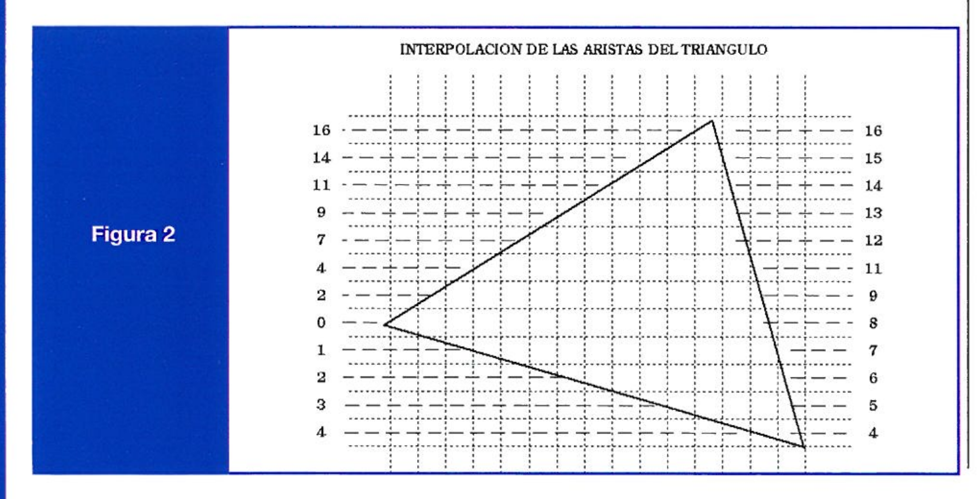
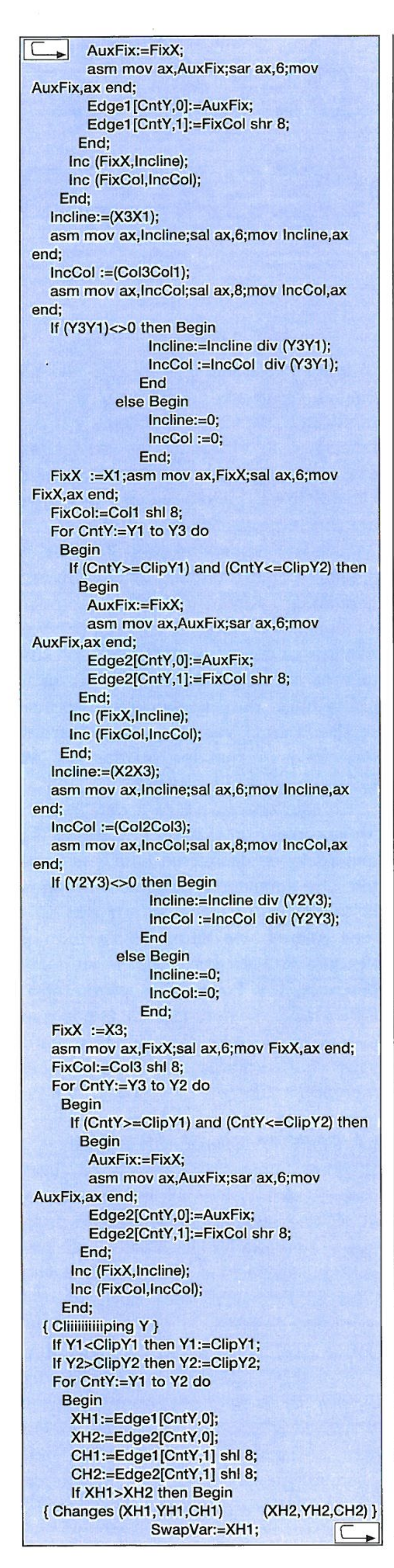


Figura 2

LISTADO 2

```
PROCEDURE GouraudFill (
SegDes:Word;X1,Y1,X2,Y2,X3,Y3:Integer;Col1,C
ol2,Col3:Byte);
Var
{ Edge arrays: }
{  0 Edge   }
{  1 Color  }
  Edge1   : Array [0..199,0..1] of Integer;
  Edge2   : Array [0..199,0..1] of Integer;
  SwapVar : Integer;
  FixX    : Integer;
  FixCol  : Integer;
  AuxFix  : Integer;
  Incline : Integer;
  IncCol  : Integer;
  CntY    : Integer;
  XH1,XH2 : Integer;
  CH1,CH2 : Integer;
  IncColH : Integer;
Begin
{ X1 to upper }
  If Y1>Y2 then Begin { Changes (X1,Y1) (X2,Y2)
}
          SwapVar:=X1;
          X1:=X2;
          X2:=SwapVar;
          SwapVar:=Y1;
          Y1:=Y2;
          Y2:=SwapVar;
          SwapVar:=Col1;
          Col1:=Col2;
          Col2:=SwapVar;
         End;
  If Y1>Y3 then Begin { Changes (X1,Y1) (X3,Y3)
}
          SwapVar:=X1;
          X1:=X3;
          X3:=SwapVar;
          SwapVar:=Y1;
          Y1:=Y3;
          Y3:=SwapVar;
          SwapVar:=Col1;
          Col1:=Col3;
          Col3:=SwapVar;
         End;
{ 12 line will be the largest ;)   }
  If (Y2Y1)<(Y3Y1) then Begin { Changes (X2,Y2)
              (X3,Y3) }
               SwapVar:=X2;
               X2:=X3;
               X3:=SwapVar;
               SwapVar:=Y2;
               Y2:=Y3;
               Y3:=SwapVar;
               SwapVar:=Col2;
               Col2:=Col3;
               Col3:=SwapVar;
              End;
  Incline:=(X2X1);
  asm mov ax,Incline;sal ax,6;mov Incline,ax
end;
  IncCol :=(Col2Col1);
  asm mov ax,IncCol;sal ax,8;mov IncCol,ax
end;
  If (Y2Y1)<>0 then Begin
            Incline:=Incline div (Y2Y1);
            IncCol :=IncCol  div (Y2Y1);
           End
        else Begin
            Incline:=0;
            IncCol :=0;
           End;
 FixX  :=X1;asm mov ax,FixX;sal ax,6;mov
FixX,ax end;
 FixCol:=Col1 shl 8;
 For CntY:=Y1 to Y2 do
 Begin
 If (CntY>=ClipY1) and (CntY<=ClipY2) then
 Begin
```





```
AuxFix:=FixX;
asm mov ax,AuxFix;sar ax,6;mov
AuxFix,ax end;
    Edge1[CntY,0]:=AuxFix;
    Edge1[CntY,1]:=FixCol shr 8;
   End;
  Inc (FixX,Incline);
  Inc (FixCol,IncCol);
 End;
Incline:=(X3X1);
asm mov ax,Incline;sal ax,6;mov Incline,ax
end;
IncCol :=(Col3Col1);
asm mov ax,IncCol;sal ax,8;mov IncCol,ax
end;
If (Y3Y1)<>0 then Begin
          Incline:=Incline div (Y3Y1);
          IncCol :=IncCol  div (Y3Y1);
         End
      else Begin
          Incline:=0;
          IncCol :=0;
         End;
FixX :=X1;asm mov ax,FixX;sal ax,6;mov
FixX,ax end;
FixCol:=Col1 shl 8;
For CntY:=Y1 to Y3 do
 Begin
  If (CntY>=ClipY1) and (CntY<=ClipY2) then
   Begin
    AuxFix:=FixX;
    asm mov ax,AuxFix;sar ax,6;mov
AuxFix,ax end;
    Edge2[CntY,0]:=AuxFix;
    Edge2[CntY,1]:=FixCol shr 8;
   End;
  Inc (FixX,Incline);
  Inc (FixCol,IncCol);
 End;
Incline:=(X2X3);
asm mov ax,Incline;sal ax,6;mov Incline,ax
end;
IncCol :=(Col2Col3);
asm mov ax,IncCol;sal ax,8;mov IncCol,ax
end;
If (Y2Y3)<>0 then Begin
          Incline:=Incline div (Y2Y3);
          IncCol :=IncCol  div (Y2Y3);
         End
      else Begin
          Incline:=0;
          IncCol:=0;
         End;
FixX :=X3;
asm mov ax,FixX;sal ax,6;mov FixX,ax end;
FixCol:=Col3 shl 8;
For CntY:=Y3 to Y2 do
 Begin
  If (CntY>=ClipY1) and (CntY<=ClipY2) then
   Begin
    AuxFix:=FixX;
    asm mov ax,AuxFix;sar ax,6;mov
AuxFix,ax end;
    Edge2[CntY,0]:=AuxFix;
    Edge2[CntY,1]:=FixCol shr 8;
   End;
  Inc (FixX,Incline);
  Inc (FixCol,IncCol);
 End;
{ Cliiiiiiiiiping Y }
If Y1<ClipY1 then Y1:=ClipY1;
If Y2>ClipY2 then Y2:=ClipY2;
For CntY:=Y1 to Y2 do
 Begin
  XH1:=Edge1[CntY,0];
  XH2:=Edge2[CntY,0];
  CH1:=Edge1[CntY,1] shl 8;
  CH2:=Edge2[CntY,1] shl 8;
  If XH1>XH2 then Begin
{ Changes (XH1,YH1,CH1)       (XH2,YH2,CH2) }
          SwapVar:=XH1;
```

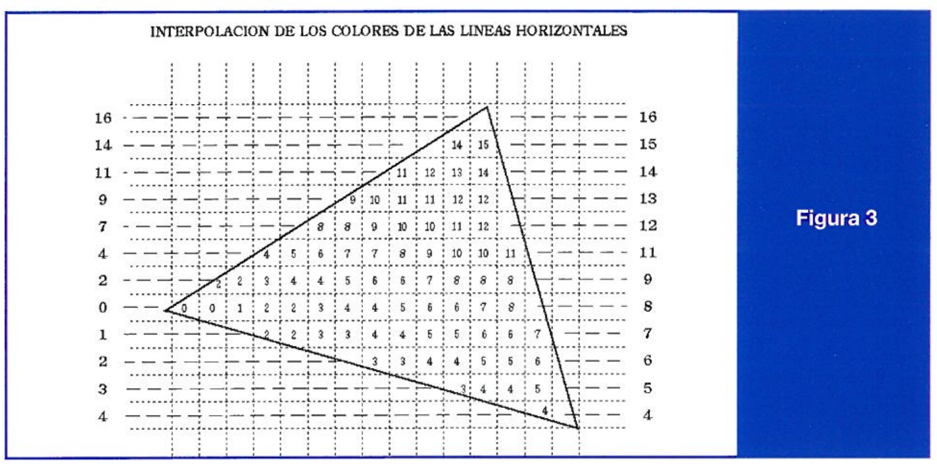
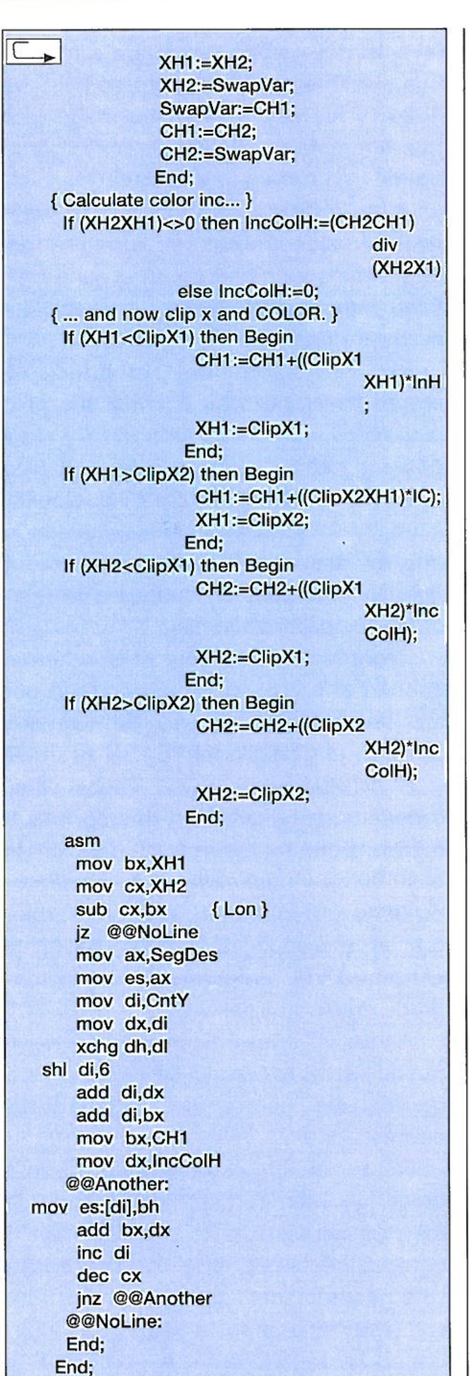


Figura 3



```
          XH1:=XH2;
          XH2:=SwapVar;
          SwapVar:=CH1;
          CH1:=CH2;
          CH2:=SwapVar;
         End;
  { Calculate color inc... }
  If (XH2XH1)<>0 then IncColH:=(CH2CH1)
                                  div
                                  (XH2X1)
            else IncColH:=0;
  { ... and now clip x and COLOR. }
  If (XH1<ClipX1) then Begin
            CH1:=CH1+((ClipX1
                              XH1)*InH
                              ;
            XH1:=ClipX1;
           End;
  If (XH1>ClipX2) then Begin
            CH1:=CH1+((ClipX2XH1)*IC);
            XH1:=ClipX2;
           End;
  If (XH2<ClipX1) then Begin
            CH2:=CH2+((ClipX1
                              XH2)*Inc
                              ColH);
            XH2:=ClipX1;
           End;
  If (XH2>ClipX2) then Begin
            CH2:=CH2+((ClipX2
                              XH2)*Inc
                              ColH);
            XH2:=ClipX2;
           End;
  asm
   mov  bx,XH1
   mov  cx,XH2
   sub  cx,bx      { Lon }
   jz   @@NoLine
   mov  ax,SegDes
   mov  es,ax
   mov  di,CntY
   mov  dx,di
   xchg dh,dl
 shl  di,6
   add  di,dx
   add  di,bx
   mov  bx,CH1
   mov  dx,IncColH
  @@Another:
mov  es:[di],bh
   add  bx,dx
   inc  di
   dec  cx
   jnz  @@Another
  @@NoLine:
  End;
 End;
End;
```

EL EJEMPLO

En el ejemplo incluido en el CD se ha sustituido el donut que venía acompañando a este curso por el pato, otra malla muy usada en el mundo de la demoscene.

Recordemos los pasos a seguir para mover un objeto en tres dimensiones con las librerías que acompañan a este artículo:

a) Crear un degradado en la paleta para dar mayor sensación de perspectiva.
b) Inicializar las variables tridimensionales de posición del observador en el eje Z (determinará el tamaño del objeto), así como los ángulos de rotación iniciales.
c) Reservar memoria para el buffer.
d) Limpiar el buffer o restaurarlo si se usa un fondo.
e) Rotar el objeto sobre los tres ángulos.
f) Proyectar el objeto en dos dimensiones, desplazando el centro de la pantalla. De esta forma se consigue una falsa sensación de movimiento espacial.
g) Ordenar las caras del objeto en orden de Z decreciente.
h) Pintar las caras visibles, en este caso con sombreado Gouraud. Usando el método de la normal.
i) Esperar el retrazo vertical.
j) Volcar el buffer a la memoria de vídeo.
k) Actualizar los ángulos de rotación.
l) Repetir el proceso desde el paso d si no se pulsa la tecla de ESC.

EL MUNDO DEL MICROPROCESADOR

Enrique de Alarcón

Actualmente existe disponible en el mercado una gran cantidad de modelos de microprocesador destinados al mundo del PC. En este artículo se detallarán todos para que el lector tenga una visión clara de qué es lo que realmente hay en la actualidad a disposición del público.

Ahora hace tan sólo 25 años, en el año 1971, se presentó en el mercado un pequeño invento que estaba destinado a marcar una nueva era. Ese invento se llamaba 4004, y fue el primer microprocesador que apareció en el mundo obra, cómo no, de una casa llamada Intel.

A partir de ese momento, las consolas de videojuegos y ordenadores domésticos se convirtieron en una realidad y un mercado de gran potencial en hardware apareció de la nada.

En ese momento, multitud de empresas empezaron a crear sus propios chips, que cada vez eran más rápidos y potentes, gracias a lo cual hizo que el tema del hardware evolucionara a una velocidad vertiginosa, haciendo que en dos años un microprocesador quedase obsoleto en cuanto a la relación precio/prestaciones.

Recordemos aquellos años ochenta, cuando el mercado estaba copado por los famosos Spectrum, de la casa Sinclair, el Commodore 64, el Atari ST y el Amiga, entre otros. Todos ellos, ordenadores basados en micros más o menos potentes (como el Z80 de la casa Zilog) y destinados sólo al mercado del entretenimiento, y que absorbieron el mercado doméstico hasta la entrada del PC en el mundo del mercado de consumo masivo.

Ahora, 25 años después del 4004, se puede ver el resultado de dicha guerra dentro del sector de ordenadores domésticos PC.

Desde que en los ochenta empezó a crecer geométricamente el mercado de los PCs domésticos (o sea, de micros compatibles x86), Intel ha creado ya seis generaciones de microprocesadores (está ultimando la séptima), duplicando prácticamente la potencia y velocidad en cada una de ellas respecto a sus predecesora. Con todo ello ha logrado hasta hoy en día estar siempre por delante de sus competidores y mantenerse a la altura de las demandas requeridas en todo momento por sus clientes (o sea, medio mundo) en lo que a potencia/prestaciones/precio se refiere.

Es en la actualidad precisamente, a finales de siglo, cuando por primera vez acaban de aparecer modelos de procesadores de su competencia que pueden rivalizar en cuanto prestaciones con sus últimos modelos de micros, lo cual puede hacer que ese preciado mercado se divida cada vez más entre diversos fabricantes y que los precios de las CPU's bajen drásticamente.

En este artículo se va a dar, tras un breve repaso de referencia a las viejas generaciones de micros Intel que existen, una detallada explicación de todos los aspectos técnicos de sus dos últimos micros, los llamados Pentium y Pentium Pro, así como los de su competencia, los PowerPC y algún otro. Para acabar, se hará referencia a lo que espera en el futuro más próximo, para saber hacia dónde está orientado el mercado actual.

LA FAMILIA 80x86 DE INTEL

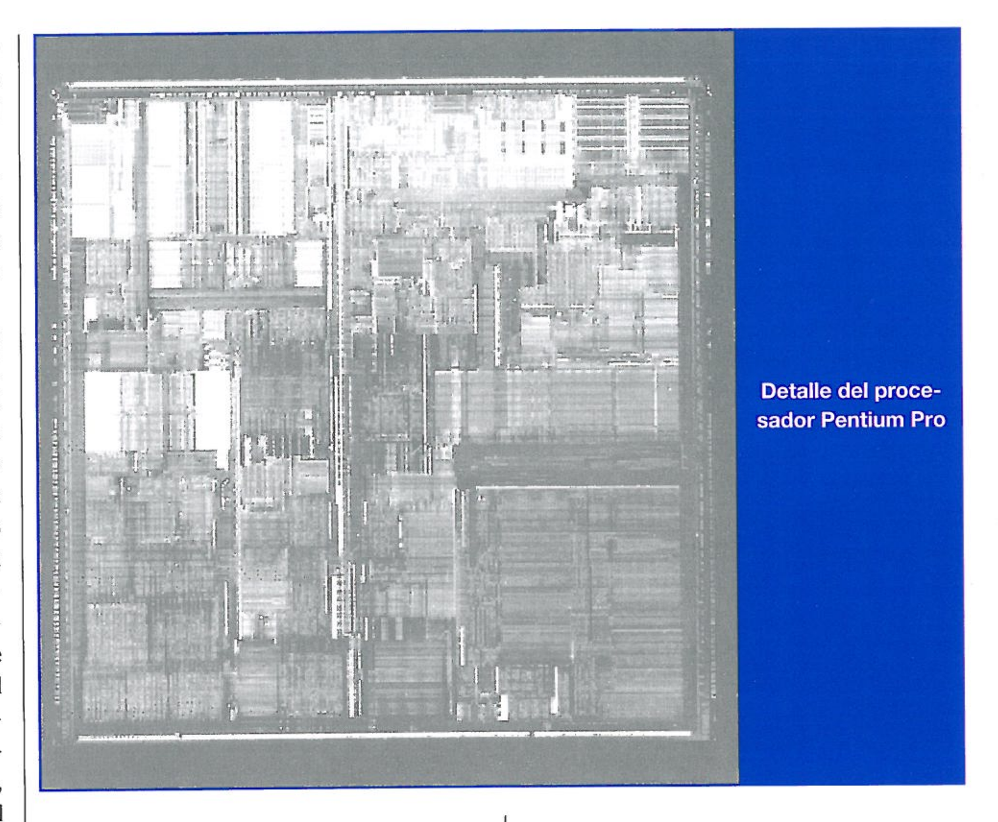
El primer procesador creado por Intel para el sector microinformático fue el 8086, que sería el primer modelo de lo que se llamaría familia 80x86. Este primer chip apareció en el año 1978 y tenía unas características muy buenas en el momento de aparecer. Era un procesador de 16 bits (cuando todos eran, como mucho, de 8), gracias a lo cual se podía direccionar la entonces enormidad de 1 Mbyte de memoria y arquitectura tipo CISC (instrucciones complejas con microcódigo asociado). Su velocidad de funcionamiento podía ser de hasta 12 Mhz.

Casi simultáneamente al 8086, apareció el procesador 8088, que era prácticamente idéntico al anterior y el cual no incorporaba nada nuevo.

Ocho años más tarde apareció el llamado 80286, que incorporaba nuevas instrucciones, pero que seguía bastante limitado en cuanto a capacidad de gestión de memoria se refiere.

En 1986 apareció lo que se podría llamar un micro revolucionario. El micro se llamaba 80386 (siguiendo el nombre familiar), y al tratarse de un chip de 32 bits tenía la capacidad física de manejar hasta 4 Gbytes de memoria (2^32 bytes). Como características generales se puede decir que poseía un set de registros de 32 bits que ejecutaba las instrucciones mucho más rápidamente que sus predecesores y que poseía una gran novedad, la posibilidad de multitarea, gracias a la incorporación de varios modos de funcionamiento del micro (modo real, real virtual, protegido...) y lo que sería fundamental para la posterior aparición del revolucionario Windows y su famosa capacidad multitarea. Además, este micro podía alcanzar ya hasta velocidades de 33 y 40Mhz, con lo que los programas a ejecutar podían ser bastante potentes.

En 1989 apareció lo que se podría considerar una versión mejorada del 80386, el 80486, que poseía una velocidad de ejecución aún más optimizada en instrucciones y un detalle muy importante, la incorporación de 8 Kbytes de memoria caché de nivel 1 (caché incorporada dentro del mismo chip) y que aumentaba su rendimiento de forma impresionante. Además de todo ello, poseía en muchos modelos, incorporado ya de serie, un coprocesador matemático que le daba la capacidad de cálculo de una calculadora científica.



Detalle del procesador Pentium Pro

El primer procesador del mundo fue el 4004 de la casa Intel

Todos los procesadores mencionados están ya fuera de uso y, aparte de algún moderno modelo 486, están todos fuera ya del mercado.

Los últimos modelos Intel que han aparecido, y que tienen copado actualmente el mercado, son los llamados Pentium (que sería como decir 80586) y el reciente Pentium Pro, que se van a detallar a continuación.

EL PENTIUM

En 1993 Intel comenzó la comercialización del llamado procesador de quinta generación al cual, para poder registrar como nombre con *copyright* (marca registrada), puso el nombre de Pentium, y con lo cual se eliminaba el problema legal de que cualquier clónico pudiese usar nombres idénticos para sus micros y confundir así a los compradores.

Este ha sido el primer procesador que ha incorporado la arquitectura interna totalmente novedosa y basada en tecnología nueva.

Este micro tiene unas dimensiones de 262mm, está diseñado a una escala de 0.8 micras usando el sistema BiCMOS (el primer modelo) y contiene 3.1 millones de transistores. Para su funcionamiento requiere un voltaje de 5 voltios.

Posee una arquitectura superescalar tipo RISC (El primero de la casa Intel) con dos unidades separadas de ejecución de 5 etapas cada una que funcionan en paralelo, con lo cual el micro puede procesar dos instrucciones simultáneamente siempre y cuando no dependan de los resultados entre sí para su proceso. Por ejemplo, no es posible procesar simultáneamente ADD AX,10 y ADD AX,BX, puesto que la segunda depende del resultado de la primera).

El Pentium posee un total de 16 Kbytes de memoria caché interna (nivel 1) divididos en dos bloques de 8 Kbytes cada uno destinados a datos y a código respectivamente, con lo cual se optimiza mucho más su eficacia (en los 486 había 8 Kbytes tanto para código como para datos, lo que daba una tasa de aciertos mucho menor).

El coprocesador matemático, el cual viene incluido en todos los modelos integrado ya de fábrica, incluye algoritmos optimizados para las operaciones más comunes, lo que hace que sea capaz de realizar cálculos de coma flotante en un sólo ciclo de reloj (sumas, multiplicaciones...). El resultado hace que funcione varias veces más rápido que el coprocesador 487.

El Pentium posee también una *Unidad de predicción de saltos* gracias a la cual, cuando se llama por segunda vez a una llamada condicional ejecuta-

da ya recientemente, la CPU no requiere ciclos extras para procesar el cálculo de salto, haciéndolo en un sólo ciclo (imaginen su utilidad en un bucle de 1 millón de ciclos que realiza continuos saltos condicionales interiores) gracias a un pequeño *buffer* que posee y donde tiene almacenados los últimos saltos condicionales que ha calculado.

El Pentium es, además, el primero que incorpora capacidades de multiproceso simétrico, con lo que se pueden crear placas base con hasta cuatro procesadores funcionando en paralelo.

En lo que a velocidades se refiere, el Pentium dispone de muchos modelos, y son los siguientes: 60Mhz, 66, 75, 90, 100, 120, 133, 150, 166 Mhz, y recientemente han aparecido a velocidades de 200Mhz, los cuales se pueden casi comparar con estaciones de trabajo de gama baja. Para tener una referencia, decir que el modelo de 166Mhz tiene unas prestaciones de 217.3 en SPECint92, 4.76 en SPECint92 y de 3.37 en SPECfp95.

OTRAS CUALIDADES DEL PENTIUM

Pese a todas las novedades que incorpora el Pentium, el micro continua siendo totalmente compatible con los procesadores anteriores 80x86 (si no, hubiese sentenciado su muerte) y por lo tanto, totalmente compatible con todo el software actual del mercado.

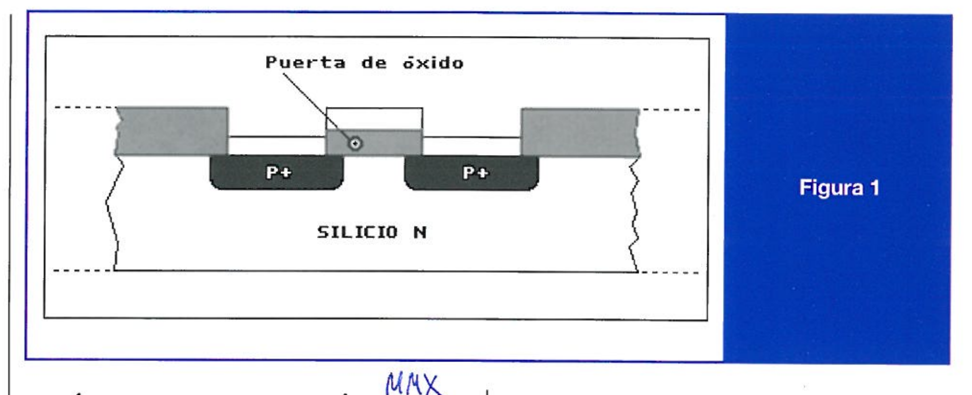


Figura 1

QUÉ ES LA TECNOLOGÍA XMM

MMX

Los procesadores que incluyan la tecnología XMM consistirán básicamente en microprocesadores optimizados en algunas instrucciones concretas y que poseerán un set extra de 57 nuevas instrucciones máquina que han sido creadas para ayudar en cálculos y operaciones de tipo multimedia, lo que hará que el micro pueda operar con datos de 64 bits y realizar operaciones de decodificación de vídeo, sonido, cálculos 3D y otros con unas pocas instrucciones/ciclos de CPU.

Dicha tecnología será incluida por Intel en todos los futuros procesadores Pentium, Pentium Pro y el mismísimo P7, que está siendo ya desarrollado.

El resultado de esto será fácilmente visibleñ. Aparecerán juegos de tipo 3D con capacidades de cálculo increibles, como es la interpolación trilineal *mip-mapping* (que usa la consola Nintendo 64), y usarán resoluciones de 1024x768 con paletas de miles de colores, todo ello con altas tasas de actualización de pantalla, debido a que dicha tecnología aumenta hasta un 400% la velocidad de ejecución según diversas fuentes, en lo que a operaciones multimedia se refiere.

El primer chip Intel de 32 bits fue el 80386

Además de esto, la aparición reciente del modelo a 200Mhz demuestra que el procesador Pentium no ha sido olvidado por Intel tras la aparición del Pentium Pro, y que por lo tanto no va a quedar obsoleto ni en desuso rápidamente por culpa de la aparición del PENTIUM PRO, ya que estos parece que prácticamente van a coexistir durante bastante tiempo en el mercado.

Además, hay noticias que afirman la próxima aparición de un nuevo modelo de Pentium llamado P55C, que será un Pentium que incluirá la llamada tecnología XMM, de la cual se habla a continuación.

EL PENTIUM PRO

La generación de micros más moderna disponible actualmente de la casa Intel es la llamada Pentium Pro. Este chip está formado por 5.5 millones de transistores (casi el doble que el Pentium) y posee una estructura interna completamente nueva respecto al Pentium. Está creado en base a un conjunto de nuevas tecnologías que globalmente han bautizado como *Ejecución especulativa*, y que supera la incapacidad del Pentium de estar ejecutando siempre dos instrucciones simultáneamente, como ocurre en el caso de que la segunda instrucción a ejecutar en paralelo dependa del resultado de la primera en ejecución, en cuyo caso debía esperarse a que acabara la primera, desaprovechando así una de las dos unidades de procesado durante varios ciclos de reloj. Para solucionar esto se ha hecho que el Pentium Pro tenga las unidades de proceso divididas en 12 etapas cada una (y no 5 como en el Pentium) y que la ejecución no se haga secuencialmente de forma lineal, sino que haya una ventana virtual de ejecución, de donde se van ejecutando las instrucciones que se puede continuamente, saltando instrucciones la segunda unidad de proceso en caso de conflicto con la que está siendo ejecutada en la primera unidad hasta encontrar una independiente, para posteriormente continuar la unidad 1 con las intermedias saltadas por la segunda y repetir el proceso indefinidamente. Como se puede deducir, este sistema necesita que el procesador use registros virtuales temporales para poder guardar los valores temporales entre ejecuciones no lineales, pero que serán totalmente trasparentes al programador.

Además, para poder acceder a varias instrucciones y direcciones de memoria a la vez es necesario que el bus soporte hasta 4 peticiones diferentes simultáneas de acceso a memoria, que son las que es capaz de realizar el Pentium Pro en momentos de proceso a máximo rendimiento.

El resultado de todas estas características técnicas es que el Pentium Pro puede llegar a estar ejecutando hasta 3

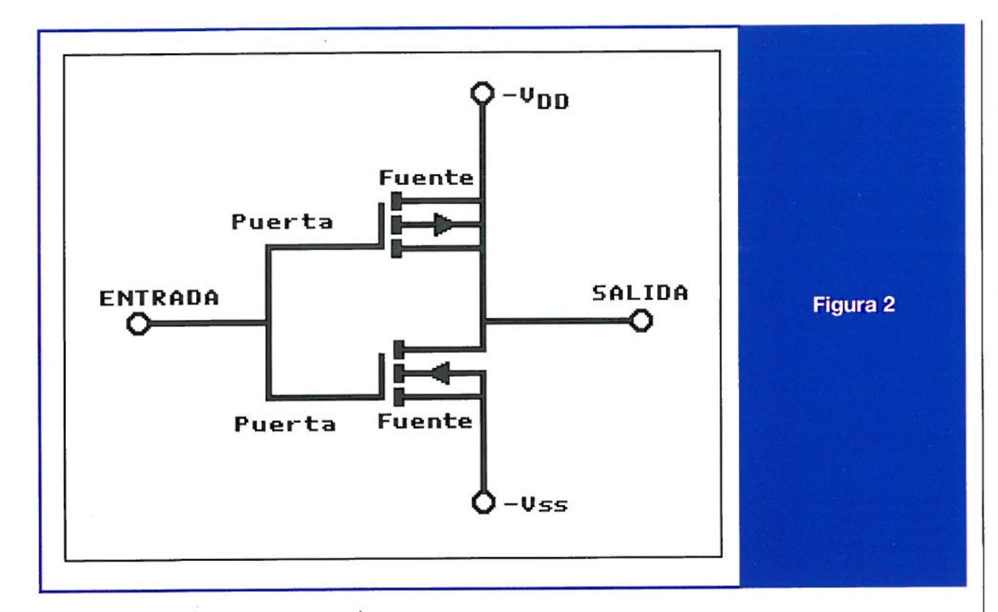


Figura 2

instrucciones simultáneamente si se une a las dos unidades enteras al coprocesador matemático, con lo cual se asegurará un 33% más de potencia que en un Pentium a igual velociad.

LA CACHÉ LEVEL 2 DEL PRO

Otra novedad curiosa del Pentium Pro respecto a todos sus antecesores es que posee la memoria caché de nivel 2 incorporada también en el propio microprocesador (actualmente incorporan 256 Kbytes y habrá modelos de hasta 512 Kbytes), lo que da como resultado que la "caja" del chip sea bastante mayor a lo normal. La ventaja de realizar esto es que la memoria Level 2 está conectada directamente a la CPU mediante un bus directo de 64 bits y, por lo tanto, funciona a su misma velocidad interna y no a los 66Mhz o similares a los que funciona el bus externo.

DATOS PARA EL COMPRADOR

El Pentium Pro, por ser un modelo más moderno que el Pentium, parte con velocidades iniciales de 150, 180 y 200 Mhz (frente a los 60 Mhz de los primeros P5), y en lo que a potencia de cálculo se refiere respecto a Pentium, comparando modelos de igual velocidad da unos resultados (Benchmark Norton SI32) de prácticamente el doble, pues si el Pentium a 200 Mhz daba 50 Benchmarks, el Pentium Pro da más de 90, con lo que la diferencia de precio queda totalmente justificada.

Pese a todo lo dicho, hay que tener un dato presente. El Pentium Pro ha sido diseñado como microprocesador para ser usado en sistemas de 32 bits, y como tal tiene un diseño optimizado para rendir al máximo funcionando en modo protegido 32 bits, ya que en modo real da un rendimiento prácticamente idéntico al de un Pentium de igual velocidad de proceso.

Así pues, si va a usar en su ordenador aplicaciones DOS y/o Windows 3.1 o Windows 95, que poseen todo o gran parte del código en 16 bits, la diferencia de precio no va a quedar justificada y valdrá más que compre un Pentium, ya que si no pagará más por resultados similares.

EL FUTURO DEL PENTIUM PRO

Al igual que pasa con el modelo anterior de la casa, el Pentium, Intel también tiene en proyecto crear una versión de Pentium Pro con las extensiones multimedia XMM, aparte de que sacará además versiones más rápidas del chip (con velocidades aún desconocidas).

Para convertirlo en un chip todavía más potente, Intel tiene además la intención de sacar próximamente una versión mejorada de Pentium Pro que incorporará una nueva caché level 2 de alta velocidad, y que estará optimizado también para funcionar mejor en modo 16 bits, lo cual es, como se ha mencionado en el apartado anterior, el único punto flojo que le queda respecto al Pentium.

LA PRÓXIMA GENERACIÓN INTEL

Mientras los consumidores piensan cuál de los micros existentes en el mercado sería mejor comprar para sus necesidades buscando la mejor relación precio/prestaciones, Intel ha anunciado ya algunas de las características de la próxima generación de microprocesadores que va a lanzar el próximo año.

Dicho procesador tiene el nombre de *Klamath* (aunque se cambiará seguramente más adelante) y tendrá una velocidad en sus versiones iniciales de 200 y 233 Mhz. Incorporarán inicialmente 256 Kbytes de memoria caché de nivel 2 (del tipo llamado *pipeline Burst-mode*) aunque también hay en proyecto crear versiones con 512 Kbytes.

Estará basado en varias tecnologías usando, por ejemplo, la ejecución especulativa, ya implementada en el Pentium Pro (detallada anteriormente), además de incorporar también las extensiones MMX multimedia, que incorporarán al parecer todos los micros Intel del futuro.

Además de todo esto, el proyecto global de este nuevo procesador incluye llegar en el futuro hasta modelos que funcionarán a velocidades de hasta 350 Mhz, con lo cual el lector se puede hacer una idea de la potencia que le espera en los micros del futuro inmediato.

LOS NUEVOS POWER PC

En marzo de 1994, Apple presentó ante el mundo unos nuevos ordenadores basados en un microprocesador nuevo, llamado Power PC, el cual era fruto de un acuerdo firmado entre las empresas Apple/IBM/Motorola pensado con el fin de crear una nueva generación de procesadores completamente nuevos no compatibles con la familia 80x86 de Intel que estarían destinados a competir con los procesadores Intel.

En este acuerdo, Motorola pondría la experiencia en la creación de microprocesadores y la tecnología MOS11, que permite crear micros más pequeños que los creados con otros métodos, y que por lo tanto salen más baratos de producir, ya que se pueden hacer más micros en una misma oblea de silicio.

Por otra parte, Apple aportaría su famoso sistema operativo System 7, tan extendido (sobre todo en los Estados unidos), además de incorporar los nuevos micros a sus sistemas Macintosh, lo que daría un mercado inicial ya hecho a los nuevos micros. El tercer firmante, IBM , aportaría la experiencia en la creación micros avanzados, la llamada tecnología POWER, siglas que provienen del inglés *Performance Optimized Whit Enhaced RISC,* y que es una arquitectura superescalar de 64 bits de gran potencia usada ya anteriormente por IBM.

El resultado final de todo ello ha sido la aparición en el mercado de una familia de procesadores llamados globalmente PowerPC, con diversos modelos adaptado cada uno de ellos a diversas necesidades, y basados todos ellos en la arquitectura RISC anteriormente mencionada. Todo ello a un bajo coste de producción.

ESTADO ACTUAL DEL POWERPC

En estos años se ha hablado de muchos problemas, retrasos y de falta de mercado, y el resultado ha sido el siguiente:

En todo el año 95, el porcentaje de ordenadores personales Macintosh vendidos con micros PowerPc (los llamados PowerMac) ha sido de tan solo un 2.6% respecto al total mundial de ventas. Así mismo, IBM sólo ha conseguido que un 0.03% de los ordenadores vendidos en el mundo en ese año fuesen PC IBM con su PowerPC.

El resultado de todo estos problemas se puede deducir fácilmente a partir de la decisión tomada por IBM respecto al tema, y es que IBM ha paralizado la producción de equipos personales que incorporan el micro PowerPC debido a su dificultad para introducirse en el mercado doméstico para poder así centrarse en productos de gama alta, como son por ejemplo las estaciones de trabajo RS/6000, lo cual paraliza cualquier posibilidad futura de una ampliación del mundo PC basado en el PowerPC de IBM.

MODELOS POWERPC

A pesar de todo lo mencionado anteriormente, la familia de micros PowerPc es bastante amplia, ya que posee diversos modelos y velocidades. A continuación se detallarán todos, explicando todas sus características técnicas que, como se verá, no son pocas.

EL POWERPC 601

El Power 601 se presentó al mundo en octubre de 1993, y fue el primero de la familia en aparecer. Su nombre comercial se simplifica como PPC601.

Este micro, pese estar basado en la arquitectura POWER de 64 bits, está creado como un procesador de 32 bits que incorpora un bus de datos de 64 bits (una especie de híbrido). Está formado por 2.6 millones de transistores y diseñado a una escala de 0.6 micras, lo que hace que tenga tan sólo unas dimensiones de 11 milímetros cuadrados. Incorpora como caché level 1 (interna) 32 Kbytes mixtos, que sirven tanto para código como para datos. Como se sabe, es un procesador basado en tecnología RISC y está dividido en tres unidades de procesamiento (unidad entera, coprocesador y direccionamiento), lo que permite que ejecute hasta 3 instrucciones simultáneamente. La llamada unidad de direccionamiento, es el equivalente a la Unidad de predicción de saltos que posee el Pentium (ya comentada) usada para detectar cambios lógicos del flujo del código.

Este chip se encuentra actualmente en el mercado funcionando a velocidades de 60, 66, 75, 80 y 120 Mhz, lo cual supera las intenciones iniciales de IBM, que tenía inicialmente en proyecto tan sólo modelos de 60 Mhz.

EL POWERPC 603

El siguiente microprocesador en aparecer fue el PowerPC 603, y fue llamado comercialmente PPC603 para seguir la misma línea de su antecesor de la familia. Este chip fue acabado en 1994, y se construyó pensando en ser un micro destinado al mercado de los portátiles. Este micro, basado, cómo no, en la arquitectura RISC POWER, está formado por tan sólo 1.6 millones de transistores (1 millón menos que el PPC601) y fabricado a una escala de 0.5 micras, por lo que tiene un tamaño menor. Además de ello, está diseñado para funcionar a menor voltaje que el PPC601, lo cual hace que consuma menos de la mitad de energía para funcionar, convirtiéndose así, por lo tanto, en un micro idóneo para ordenadores portátiles, que es lo que precisamente buscaban sus diseñadores.

Otras diferencias respecto al PPC601 es que posee 16 Kbytes de caché en vez de 32Kbytes divididos en dos bloques separados de 8Kbytes, uno para código y otro para datos, lo que le da un mayor rendimiento global (y causa principal de que el chip esté formado por tan pocos transistores respecto al PPC601).

Los micros PowerPC son fruto de la alianza IBM/Apple/Motorola

Para ampliar sus posibilidades de ahorro de energía el chip dispone, además de su especial diseño, de tres modos de funcionamiento activables por software destinados a que se pueda dejar el micro en modo de ahorro de energía en los momentos de menos actividad de proceso (una especie de *Standby*).

En el resto de características, es idéntico al 601 y es, al igual que dicho micro, capaz de procesar un máximo de 3 instrucciones por ciclo de reloj, puesto que posee también tres unidades de proceso. Y en lo que a velocidad de proceso se refiere, se puede decir que hay modelos funcionando en portátiles que llegan también hasta los 120 Mhz.

Para acabar, comentar un dato importante, y es que el PPC603 es totalmente compatible con el PPC601, por lo que puede ejecutar todos los programas de aquél.



EL POWERPC 604

El tercer micro de la familia PowerPC es el micro PowerPC 604, o PPC604, y mientras que el PPC603 estaba pensado para funcionar como micro de portátiles, el PPC604 ha sido creado con el fin de ser usado en estaciones de trabajo y servidores.

Es un micro hecho también a una escala de 0.5 micras y el primer detalle que delata su mayor potencia de proceso es que está diseñado para poder ejecutar hasta un máximo de cuatro instrucciones por ciclo de reloj, lo que le da un 30% más de potencia de cálculo respecto a un PPC601 funcionando a igual velocidad.

Otro dato importante es que está pensado para funcionar a mucha más velocidad, y ello se puede confirmar por la noticia dada por Motorola de presentar próximamente un PowerPC 604 que funcionará a 300 Mhz, aunque existen ya en la actualidad modelos que llegan a los 200 Mhz y que dan un rendimiento de 225 en la escala SPECint92, 6 en la SPECfp92 y de 5 en la escala SPECfp95.

EL SISTEMA OPERATIVO DEL POWERPC

Como ya se ha dicho, la familia PowerPC no es compatible con la familia 80x86 de Intel, lo cual quiere decir que los ordenadores PowerPC tendrían que disponer de su propio sistema operativo.

El AMD-K5-PR100 de AMD es un micro clónico compatible con el Pentium 120 Mhz de mejor rendimiento

IBM, cuando presentó el proyecto PowerPC en 1991 junto a sus dos aliados, tenía la intención de tener disponible una versión de OS/2 nativa para PowerPC al mismo tiempo que apareciesen los primeros ordenadores PPC6xx. Pero la realidad fue mucho más triste, y es que hasta Enero de 1996 no ha aparecido una versión beta inicial disponible a pequeña escala de dicho sistema operativo, lo cual hace pensar que también habrá influido negativamente en la venta de sus plataformas, ya que es de suponer que IBM habrá perdido potenciales compradores al haber retrasado el sistema operativo durante dos años, puesto que los clientes no habrán esperado ese tiempo y se habrán pasado a la plataforma Intel/Microsoft (Pentium/Windows NT).

En 1997 aparecerán micros Intel que llegarán como mínimo a los 233 Mhz

EL POWERPC 620

El PowerPC 620 o PPC620 está destinado a competir en potencia con los procesadores más modernos Intel, como el Pentium Pro, y ha empezado a circular en el mercado en pequeñas cantidades desde mediados del presente año.

Ha sido creado pensando en ser usado también como procesador de estación de trabajo y otros con gran demanda de potencia de proceso, y la principal novedad que incorpora este micro es que posee una arquitectura interna RISC POWER totalmente de 64 bits, lo que le da una gran potencia en lo que a manipulación de datos se refiere.

En lo que a velocidad se refiere, los modelos iniciales corren a una velocidad de 133 Mhz, aunque es posible que existan modelos ya con velocidades superiores.

OTROS MICROS

Aparte de la familia PowerPC, existen otros micros en el mercado de reciente aparición que son lo suficientemente potentes como para competir con los últimos micros Intel, y algunos de los cuales son incluso compatibles a nivel de código, pudiendo ejecutar programas escritos para micros de la familia 80x86 lo que les da, en caso de salir más baratos, muchas posibilidades de mercado. A continuación se detallan los principales que han aparecido.

LA FAMILIA K DE AMD

AMD presentó a principios de 1995 un proyecto según el cual iba a crear una nueva familia de micros de altas prestaciones, que se llamaría serie K. Dichos procesadores estarán formados por un total de 4 micros (en principio) que se llamarán K5, K6, K7 y K8, y que irán apareciendo progresivamente hasta el año 2001 aproximadamente, momento en que aparecerá el último de la familia (K8).

De momento, AMD acaba de presentar al mercado ya el primer procesador de dicha familia, el llamado AMD-K5-PR100, que compite en prestaciones al Pentium de 100 Mhz de Intel. Este micro

está formado por 4.3 millones de transistores, ha sido realizado a una escala de 0.35 micras y se fabrica en tecnología CMOS. Al igual que casi todos los micros más modernos de la actualidad, está diseñado con una arquitectura superescalar (llamada en este caso tecnología K86) y como dato muy importante destacar que es totalmente compatible con todo el set de instrucciones de la familia 80x86 de Intel.

Del resto de micros de la familia que están por aparecer, se puede decir que el K6 está desarrollado también en 0.35 micras, posee 6.5 millones de transistores y se supone que estará disponible en el mercado como mínimo a finales de 1997.

El K7, que aparecerá como mínimo en 1998/1999, llegará a los 400 Mhz, estará formado por unos 10 millones de transistores y se fabricará a escala de 0.18 micras.

Los planes para el último de la familia, el K8, indican que poseerá una velocidad de 600 Mhz y que estará formado por al menos 20 millones de transistores, aunque dicho modelo no se va a ver hasta el próximo siglo (para lo cual no falta tanto) y por lo tanto es mucho adelantar sus características.

EL 6x86P200 DE IBM/AMD/CYRIX

Las empresas IBM, AMD y CYRIX, eternas competidoras de Intel en lo que a microprocesadores se refiere, se han unido para crear el chip 6x86P200, el cual acaba de ser presentado hace pocos meses al público, y tiene según sus desarrolladores un rendimiento global superior incluso al del moderno Pentium de 200 Mhz de la casa Intel, lo que es consecuencia de haber dotado a este chip de un bus externo que funciona a 75 Mhz, o sea, más rápido que el bus externo del micro Intel, que funciona a 66 Mhz.

Este micropocesador ha sido diseñado por CYRIX, y está siendo fabricado por IBM a una escala de 0.44 micras y basado en un método de fabricación CMOS de 5 capas. En sus versiones iniciales este micro funciona a 150 Mhz, aunque es de esperar que ésta velocidad sea aumentada en los modelos del futuro inmediato.

EL 6x86P200 DE IBM/AMD/CYRIX posee un rendimiento global superior al Pentium 200Mhz

MICROS DE ALTA POTENCIA

A partir de estos modelos y velocidades, lo que queda en el mercado ya corresponde a micros de empresas que desarrollan micros de alta potencia destinados a estaciones gráficas, servidores y otros. Normalmente todos estos chips se basan en tecnología RISC, y son a su vez chips de 64 o 128 bits.

Por ejemplo, Digital, una empresa muy conocida, está vendiendo desde principios de este año dos nuevos procesadores, los llamados RISC ALPHA 21164, que parten en sus versiones iniciales de una velocidad de 200, 300, 433 y 500 Mhz, y que se fabrican a escala de 0.35 micras. Dicho microprocesador, en su versión de 500 Mhz, posee un rendimiento de más de 500 SPECint92, más de 11 SPECfp92 y más de 17 SPECfp95, siendo así el chip más potente que hay en el mercado. Otros micros también de gama alta son, por ejemplo, el PA8000 y el UltraSPARC (de menor potencia que el PowerPC).

Hay proyectos para el 2001 de micros que funcionarán a 600 Mhz

COMPARATIVA

Quizás la velocidad y prestaciones de los microprocesadores de los que se ha hablado no parecen nada asombroso, ya que estamos acostumbrados a oír cada mes noticias de nuevos micros más potentes y sofisticados, pero a continuación se detallarán un poco como ejemplo las características de un micro llamado 6800 que sacó la casa Motorola hace muchos años y con las cuales se verán los cambios que han sufrido los micros en poco más de una década:

Micro 6800 de la casa Motorola.
Procesador 8 bits.
Velocidad de 1 Mhz.
Juego de 72 instrucciones.
Siete modos de direccionamiento.
Capacidad para direccionar 64 Kbytes.

Casi seguro que el lector, descontento hasta ahora con las prestaciones de los micros existentes en el mercado hoy en día, debe haber cambiado casi por arte de magia, y es que no es para menos.

RESUMEN

En este artículo se ha dado un largo repaso a la historia de los microprocesadores dentro del mundo PC. Dentro del mismo se ha explicado la historia de los micros desde sus inicios en 1978, y se han repasado superficialmente todos los principales modelos aparecidos en estos veinte años hasta llegar a las últimas generaciones de micros, que son los que realmente importa conocer más a fondo.

Puesto que son los que seguramente se comprarán al adquirir un Ordenador Personal , se han detallado los principales modelos que existen en el mercado y dado todas las informaciones disponibles sobre las características que tendrán los micros que aparecerán en el futuro inmediato (e incluso los del próximo siglo). Con todo ello, el lector está ahora en condiciones de poder elegir fácilmente qué micro es el que más le conviene adquirir según sus necesidades como usuario.

BIBLIOGRAFÍA

- Nuevas tecnologías, Biblioteca de Electrónica/Informática. Orbis. Marcombo.
- 8088-8086/8087 Programación ENSAMBLADOR en entorno MS-DOS, Miguel Ángel Rodríguez Roselló. Anaya Multimedia.
- Programación del 80386/387. Manual de referencia y técnicas avanzadas de programación para diseñadores de sistemas, programadores y usuarios avanzados. John H. Crawford/Patrick P. Gelsinger. Anaya Multimedia.

FUNCIONES GRÁFICAS EN MODO X

Santiago Romero y Miguel Cubas



A continuación se verán una serie de procedimientos con los que se completa la librería personal que cada lector debe poseer para el desarrollo de aplicaciones gráficas, demos y juegos. Hay muchas funciones gráficas que se podrían explicar (líneas, dibujo de polígonos, escalado de imágenes, etc...), pero al diseñar este artículo hemos de ceñirnos al modo X, pues la programación gráfica de este tipo de efectos ya está contemplada en otras secciones de la revista.

Por lo tanto, en el presente texto se comentarán únicamente aquellas primitivas (líneas, *buffers*, *sprites*, etc...) cuya comprensión implique que el lector sea capaz de desarrollar sus propias rutinas para otros efectos. Esto quiere decir que dibujar líneas horizontales o verticales es mucho más que lo que se obtiene en pantalla, pues significa entender la manera de organizar algunos algoritmos dentro de los modos *unchained*.

Queda claro que las rutinas que se desarrollarán actúan directamente sobre la pantalla de la VGA de una manera didáctica. Para aplicaciones que requieran elevadas velocidades, todos los cálculos deberán realizarse en una pantalla virtual de, por ejemplo, 320x200 y volcarse a la Vídeo-RAM. Para ello se desarrollarán funciones de vuelque de *buffers* y segmentos de memoria, de manera que al dibujar en estos *buffers* (lineales) se podrán usar las funciones normales de dibujo en modo estándar 13h. De esta manera, se puede usar cualquier función de dibujo (elipses, círculos, gráficos 3D, etc...,) que ya se posea dentro de los programas en Modo X.

DIBUJO DE LÍNEAS VERTICALES

Este tipo de línea es el más fácil y rápido de plasmar en pantalla, debido al sistema de planos de la VGA. Supongamos un modo X de 320x200 con organización de 1x4 pantallas (modo X por defecto), como se puede ver en la figura 1, donde también puede observarse la ventaja que se presenta al dibujar la línea verticalmente: todos los puntos que forman la línea tienen la misma coordenada X, por lo que todos están en el mismo plano (plano=1 << (x&3), ver artículo 1º).

Al tener todos los puntos de la línea en el mismo plano, sólo hay que calcular el plano en el que cae el primer punto y escribir los diversos pixels de la línea hasta llegar a la coordenada Y2. Con este tipo de líneas se evita cambiar de plano para cada punto a dibujar, hecho en el que radica su velocidad.

Veamos el algoritmo que describe estas acciones:

Dibujo de una línea vertical entre (X,Y1) y (X,Y2):

Calcular plano para (X,Y1).
Seleccionar plano de la VGA.
Calcular el offset del punto (X,Y1).
Hacer registros ES:DI = Offset calculado.
Poner color en registro AL.

Mientras no se llegue a Y2:
  Poner punto en ES:DI.
  Incrementar DI en 80 (sig. punto).

En el listado 1 puede verse el código C y Ensamblador resultante de este algoritmo. Este es bastante rápido y compacto, aunque debe hacerse notar que en otras resoluciones virtuales no habría que incrementar en 80 el valor

En esta última entrega se terminarán de perfilar los conocimientos sobre el modo X con rutinas que permitirán por fin completar la base para crear nuestros propios programas.

del registro DI, sino que habría que tener en cuenta la anchura de la pantalla, según el modo de pantalla actual (2x2 pantallas=160 bytes y 4x1=320 bytes horizontales), como se explicó en el artículo dedicado a las resoluciones virtuales que adoptaba la VGA en modo X. El hecho de que se incremente DI en 79 y no en 80 es debido a que *stosb*, que coloca el valor que contenga AL en la posición de memoria especificada por ES:DI, ya incrementa DI en una unidad tras escribir el byte, por lo que se deberá saltar en este caso 79 bytes más para posicionarse en el siguiente punto de la línea vertical. Todas las funciones que se verán a continuación están desarrolladas para el modo 13X en sistema 1x4 pantallas, pero son fácilmente adaptables a cualquier otro modo que se vaya a utilizar en los programas.

LA IMPORTANCIA DE ESTAS PRIMITIVAS

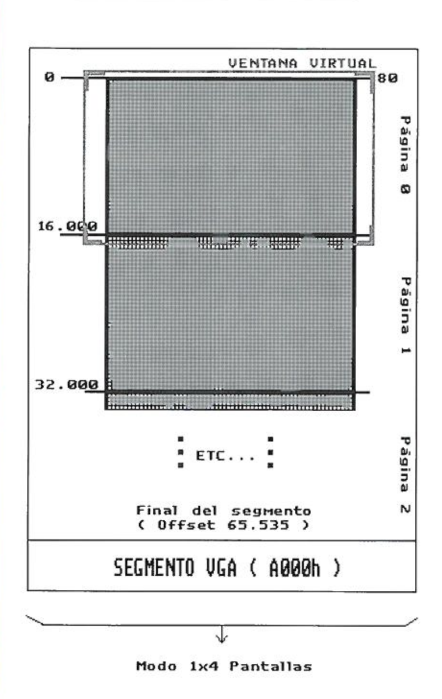
Probablemente muchos lectores se estarán preguntando de qué sirve saber dibujar líneas verticales. Su importancia se verá en el momento en que se intente desarrollar una rutina para volcar *buffers*, *sprites* o segmentos completos a la memoria de la VGA, que ya no es lineal, sino planar.

Si se trata de desarrollar una rutina simple para crear una línea horizontal directamente en pantalla (como en la figura 1), nos encontraremos con que cada punto de la línea cae en un plano distinto de la VGA. Este problema puede resolverse de dos formas: la primera de ellas sería calcular el plano con que se corresponde cada punto, cambiarlo mediante los registros de la VGA y dibujar el punto en memoria. Esto significa que para una línea de 50 pixels de ancho se realizan 50 cambios de plano, con la consecuente pérdida de velocidad.

El otro método, más laborioso, consiste en activar el plano 0 y dibujar los puntos de la línea que caen en el plano 0, haciendo lo mismo después con el plano 1, 2 y 3.

Tantos son los factores a tener en cuenta que se puede deducir que en cuestiones de velocidad la rutina de dibujo de líneas verticales es mucho más rápida que la de líneas horizontales.

DIBUJO DE LINEAS EN MODO X



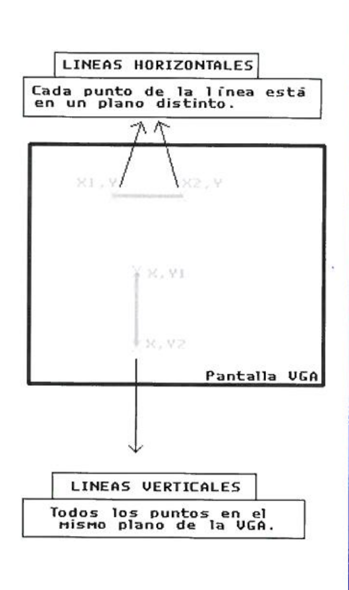


Figura 1. Dibujo de líneas en Modo X.

EL VOLCADO DE *BUFFERS* A LA VÍDEO-RAM

Veamos ahora para qué ha servido la anterior introducción al dibujo de líneas horizontales y verticales. Debe quedar claro que el volcado de *buffers* a la vídeo-RAM es una de las rutinas más importantes que se puedan crear. Mediante esta rutina puede evitarse el engorroso (y más lento) método de programación de planos y dibujar directamente sobre *buffers* temporales que más tarde se volcarán a la pantalla. Esta es, pues, una función fundamental para todos los programadores en Modo X cuyos programas requieran gran velocidad (demos, juegos, etc...).

VUELQUE POR LÍNEAS VERTICALES

Analicemos ahora el problema que se presenta al intentar volcar, como representa la figura 2, un segmento de memoria convencional que emula una pantalla de 320x200 pixels sobre la pantalla del modo 13X.

Si se estuviera bajo el modo estándar 13h, la solución se basaría en un bucle que volcara todos los bytes desde Segmento hasta 0A000h, como describe el siguiente pseudocódigo (80x086):

Mover 320 bytes desde DS:SI a ES:DI:

DS:SI = Segmento:0000
ES:DI = 0A000:0000
CX = 64000 (320x200)
REP MOVSB

Tratándose de un modo *unchained*, nos encontramos de nuevo con el problema del cambio de plano. Cada punto del *buffer* tendrá una localización en pantalla con su correspondiente plano, presentándose así dos posibilidades para volcarlo. Por una parte se puede tratar el segmento como una serie de líneas horizontales y, mediante un bucle, volcarlas a pantalla. La otra opción es considerarlo como una serie de líneas verticales, con lo que se descubre la utilidad del análisis anterior sobre dibujo de líneas que tan obsoleto parecía. Esta opción está desarrollada dentro del listado 2, y está en gran parte en C para facilitar su comprensión, aunque por supuesto no será así en la rutina incluida en la librería.

EL ALGORITMO MÁS RÁPIDO

Puede ser que con el apartado anterior el problema quede resuelto, pero los programadores en ensamblador saben que todos los algoritmos se pueden optimizar, así que con una rutina tan importante como ésta hay que rizar el rizo y para esto se dispone de una rutina más rápida, con sólo 4 cambios de plano, que los más avispados ya habrán intuido.

Para conseguir tan sólo 4 cambios de plano bastaría con activar el plano 0 y volcar en pantalla todos los puntos del *buffer* que recaigan sobre el plano 0, repitiendo el proceso para los planos 1, 2 y 3.

Esto lo podemos hacer mediante un simple bucle, ya que se sabe que si el punto 0 cae en el plano 0, el 1 sobre el 1, etc..., entonces el punto 4 corresponderá de nuevo al plano 0. Veámoslo en seudocódigo para aclarar dudas:

DS:SI = Segmento
ES:DI = 0
Activar Plano 0
Repetir 16.000 veces (64000/4)
AL = DS:SI ->Se lee un byte del segmento virtual
ES:DI = AL ->Lo escribimos
DI = DI+1 ->x = x+4
SI = SI+4 ->siguiente punto
Repetir para Planos 1, 2 y 3

Como puede apreciarse en el algoritmo, se activa el plano 0, se lee un byte del *buffer* virtual (AL = DS:SI), se escribe en pantalla (ES:DI=AL) y se pasa al siguiente punto que corresponda al plano 0 (DI+=1 y SI+=4). Repitiendo esto para 320x200/4=16000 bytes se vuelcan a pantalla, en el plano 0, todos los puntos del *buffer* que se deberían corresponder con este plano. Haciendo esto para el resto de los planos se obtiene en pantalla el *buffer* que se pretendía volcar y todo esto, aunque parezca más complicado que la anterior rutina, reduce el acceso a puertos de 320 a 4 llamadas. Todo un lujo de vuelque por software, que se puede ver traducido a código C y ASM en el listado 3.

Hay que decir también que aunque este volcado entraña gran dificultad en cuanto a la comprensión del algoritmo para los programadores de alto nivel (C/Pascal/etc..), cuando se entiende permite ya desarrollar funciones de dibujo de *sprites* por líneas verticales, bloques y otros tipos de gráficos, como puede verse en el listado 4.

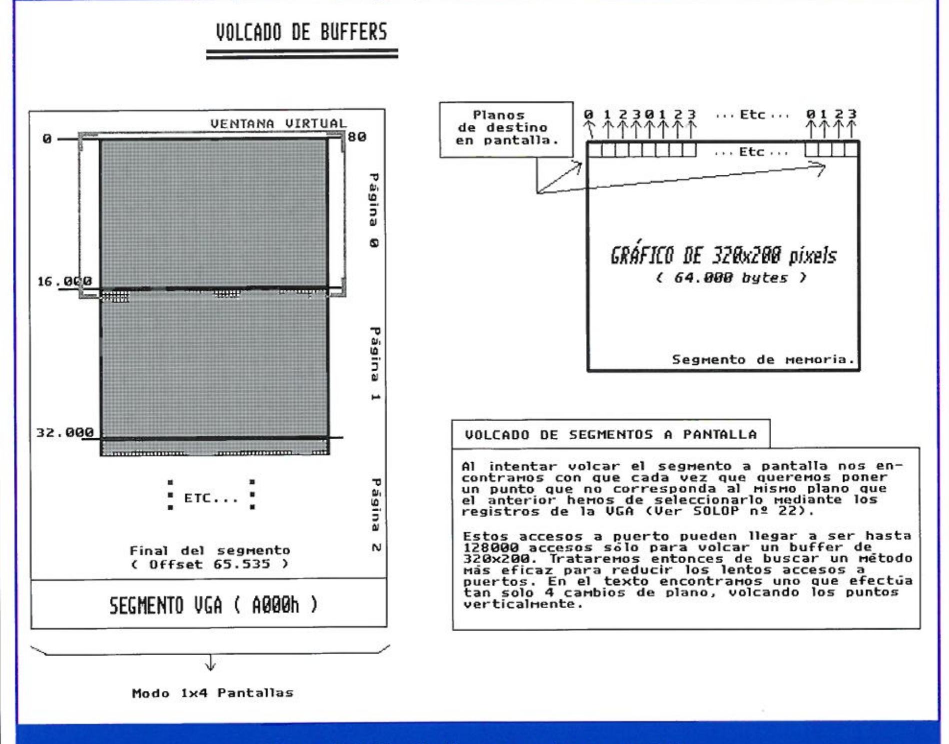


Figura 2. Volcado de segmentos a pantalla.

DIBUJO DE *SPRITES*

Considerando los *sprites* como *buffers* de un ancho y alto más reducidos, se pueden adaptar las rutinas de volcado para dibujar bloques gráficos en pantalla. En el listado 4 aparece la rutina *DrawSpriteX()*, realizada gran parte en C con el objetivo de ser fácilmente comprensible. Esta rutina resulta rápida al estar basada en el dibujo de líneas verticales del *sprite*. No obstante, la manera más rápida de dibujar *sprites* en Modo X es almacenar estos por planos. Primero los datos del plano 0, luego los del plano 1, etc... y volcarlos así en pantalla. Veamos esto con más detenimiento para realizar una rutina rápida de impresión de *sprites* en Modo X 1x4.

LA VGA A FONDO

Ya que se trató en anteriores artículos el tema de los registros de la VGA, y que sólo se utilizaron para la programación del Modo X, se va a dejar un poco de lado un tema tan interesante como es el Modo X para adentrarnos más en los registros de la VGA, que proporcionan una amplia gama de posibilidades con las que se pueden crear nuevos efectos visuales.

Este apartado se basa en su mayoría en la manipulación de los registros, ya que éstos permiten una cantidad inagotable de posibilidades. Los registros aparentemente resultan inútiles, pero de ellos se pueden extraer multitud de efectos interesantes. Sólo hay que utilizar la imaginación y experimentar con ellos. En un principio se pueden manipular realizando pruebas con todos los controladores, pero hay que tener cuidado con los registros de *Timing* (registros CRTC 0-7) ya que con una utilización continuada de valores inapropiados puede dañar el monitor (el monitor empieza a pitar), llegando incluso a estropearlo por completo si éste no se logra apagar a tiempo. El resto de registros no son peligrosos, lo más que pueden llegar a hacer es bloquear el monitor.

Para comenzar con las aplicaciones de los registros, este apartado tratará en primer lugar el registro 3C2h, llamado *Miscellaneous OutPut Register*. A simple vista no parece un registro al cual se le puede sacar partido. Tiene

LISTADO 1

```
void DrawVLine( int x, char y1, char y2, char
color )
{
char alto;
unsigned int offs;

 alto = y2-y1;                   /* y1 debe ser < y2 */

 OutPortb( SEQU_ADDR , 0x02 );              /*
seleccionar plano */
 OutPortb( SEQU_ADDR+1, 0x01 << ( x & 3 ) );

offs = (y1<<6) + (y1<<4) + (x>>2);          /* offset
*/

asm {
        mov ax , 0xa000
        mov es , ax
        mov di , [offs]    /* ES:DI = (X,Y1) */
        mov al , [color]   /* AL = Color */
        mov cl, [alto]     /* CL = Altura */
    }
VLineLoop:
asm {
        stosb              /* ES:DI = AL (y DI++) */
        add di, 79         /* siguiente punto */

        dec cl            /* comprobamos si es */
        jnz VLineLoop       /* el último punto */
    }
}
```

Dibujo de líneas verticales.

LISTADO 2

```
void FlipSegmentC( unsigned int Segm )
{
unsigned int offs;
int f,f2;

 asm push ds
 asm mov ax, 0a000h
 asm mov es, ax          /* ES:DI = VGA */
 asm mov ds, [Segm]    /* DS:SI = Segm */
 asm xor si, si

 for( f=0; f<320; f++ )   /*320 líneas verticales */
 {

  offs = f>>2 ;
  asm mov di, [offs]              /* DI = (x/4) */
  asm mov si, [f]                 /* SI = x */
  OutPortb( SEQU_ADDR , 0x02 );
  OutPortb( SEQU_ADDR+1, 0x01 << ( f & 3 ) );
/* Poner plano */

    for( f2=0; f2<200; f2++ ) /* volcar 200 pixels */
    {
      asm movsb
      asm add di, 79
      asm add si, 319
    }

 }
 asm pop ds
}
```

Volcado de segmentos por líneas verticales.

LISTADO 3

```
void FlipSegmentASM( unsigned int Segm )
{
asm {
        push ds
        mov ds, [Segm]
        xor si, si              /* DS:SI = Segm */
        mov ax, 0a000h
        mov es, ax
        xor di, di              /* ES:DI = VGA */
        xor bx, bx               /* BX = 0 (Plano) */
    }

CuatroPlanos:
asm {
        mov dx, SEQU_ADDR
        mov al, 2               /* selección de plano */
        out dx, al              /* OutPortb(
SEQU_ADDR, 2 ); */

        xor ax, ax
        inc ax                  /* AX = 1 */
        mov cl, bl               /* CL = plano */
        shl ax, cl              /* AX = 1 << plano */

        inc dx
        out dx, al              /* OutPortb(
SEQU_ADDR, AL ); */

        xor di, di              /* offset inicial 1er punto
*/
        mov si, bx
        mov cx, 16000             /* repetir 16000
veces */
    }

BuclePlano:
asm {
        mov al, ds:[si]         /* AL = DS:SI */
        mov es:[di], al         /* ES:DI = AL */
        inc di                 /* Ste. punto de pantalla
*/
        add si, 4               /* Ste. punto del buffer
*/

        dec cx
        jnz BuclePlano          /* Repetir 16000
veces */

        inc bl
        cmp bl, 4               /* Repetir 4 veces (pla-
nos) */
        jb CuatroPlanos

        pop ds
    }
}
```

Volcado de segmentos por cambios de plano.

apariencia de poca utilidad que tan sólo el hardware interno puede utilizar, pero eso no es cierto. Tiene más utilidades de las que uno podría imaginar.

En primer lugar, este registro es capaz de activar o desactivar la polaridad mediante los bits 6-7. El bit 6 es capaz de poner negativa o positiva la polaridad horizontal del retrazo, y el bit 7 hace lo mismo, pero con la polaridad vertical del retrazo. El bit 5 selecciona la página para el direccionamiento par/impar. En este caso, si el bit 5 está a 1 indica que en todos los planos se ocupan las direcciones impares. De lo contrario, si está a 0 nos ocuparan todas las direcciones pares. Los bits 2-3 se encargan de la selección del reloj de vídeo que determina la resolución horizontal (a través de la frecuencia de pixels). Si se activa el bit 1 dará acceso a la RAM de la VGA por parte de la CPU, y si se desactiva no será posible ver nada en pantalla, ya que no existe acceso alguno.

Es aconsejable ver los ejemplos que aparecen en el apartado correspondiente al Modo X del CD de la revista. También puede verse como ejemplo el listado 5, que indica la manera de programar, a partir del modo 13h, una resolución vertical de 480 líneas.

Esta ha sido la aplicación al registro *Miscellaneous OutPut Register*, cuya utilidad parecía escasa.

Como anteriormente se ha dicho, la mayoría de los registros son útiles para programar efectos gráficos. A continuación se explicará el funcionamiento de otro registro un poco más conocido por las aplicaciones que ha tenido en este curso de Modo X. El registro 09h del *CRTC Controller* (*Max Scan Line Register*) tiene utilidad a la hora de programar los efectos gráficos debido al efecto que produce. Este registro es capaz de duplicar las líneas en un modo gráfico. En caso de estar cargado el modo 13h de 320x200, este registro puede duplicar cada línea con tal de tener 320x400 líneas, cada una duplicada. Debido a su duplicación sólo se podrán ver en pantalla las 100 primeras líneas del gráfico original. Por lo tanto, la resolución resultante no es más que de 320x100 pixels en pantalla.

Esto quiere decir que éste registro se basa en la repetición de cada línea, pero no sólo la duplica, sino que tiene la posibilidad de repetir una misma línea un máximo de 15 veces, dependiendo de su programación. El resultado es que si, por ejemplo, hay que crear un plasma de 160x100 que se dibujará en 2x2 pixels, seleccionando una duplicación de 2 líneas, tan sólo habrá que dibujar los pixels como 2x1 para que sea la VGA la que doble las líneas horizontales.

El funcionamiento de este registro es muy sencillo. Tan sólo hay que leer de este puerto (*Max Scan Line Register*) un byte para poder activar el bit 7 y activar el modo *Double Scan* (modo de duplicar líneas). Una vez activado el bit



LISTADO 4

```
void DrawSpriteX( char *SprArray, int x, int y,
char ancho, char alto )
{
unsigned int offs;
int f,f2;

 asm mov ax, 0a000h
 asm mov es, ax

 for( f=0; f<ancho; f++ )
 {
  offs = (y<<6) + (y<<4) + ((x+f)>>2 );
  asm mov di, [offs]
  OutPortb( SEQU_ADDR , 0x02 );
  OutPortb( SEQU_ADDR+1, 0x01 << ( (x+f) & 3 )
);

   for( f2=0; f2<alto; f2++ )
   {
     _AL  = SprArray[(f2*ancho)+f];
     asm stosb
     asm add di, 79
   }
 }
}
```

Dibujo de sprites por líneas verticales.

LISTADO 5

```
void Set480Lines ( void )
{

InitModo13H();                  /* modo 320x200x256
*/

asm {
    mov     dx , 0x3CC  /* registro de lectura del
*/
                    /* Miscellaneous O. R.      */
    in      al , dx
    mov     dx , 0x3C2      /* registro de escritura
*/
                        /* del Misc. O. R.      */
    and     al , 00111101b
    mov     ah , 11000010b /* bits 6-7 activados
*/
                    /* bit 1 regula el acceso a la RAM de
*/
                    /* la VGA.                  */
    or      al , ah
    out     dx , al
    }

}
```

Obtención de 480 líneas de resolución vertical.

LISTADO 6

```
/ ** LISTADO 6: Duplicación de líneas en
modo 13h **/

void main()
{
SetModo13H();

asm {
    mov     dx , 0x3D4  /* CRTC Controller
*/
    mov     al , 0x09    /* Registro 09h         */
                        /* MaxScanLine Register */
    out     dx , al
    inc     dx
    in      al , dx
    and     al , 01110000b
    add     al , 129  /*10000001b - activo bits 7
y 0*/
    out     dx , al   /*Bit 0 indica repetir una vez
*/
    }           /* Bit 7 activa el modo de repetición
*/
}
```

Duplicación de líneas en modo 13h.

7 se tiene que indicar el número de veces que se repetirá una sola línea con el fin de lograr una nueva resolución. Se puede repetir una línea como máximo 15 veces debido a que son los bits 0-3 los que se encargan de indicar dicha función.

Si se quiere programar la repetición de líneas, el bit 7 siempre debe estar activado (puesto a 1). Si el bit 7 no está puesto a 1, la VGA toma su forma extendida de 400 líneas (su forma real en este modo gráfico, ya que la VGA no reconoce ningún modo de menos de 350 líneas. Los modos de 200 líneas son generados por 400 líneas físicas). Si el bit 7 se pone a 1 se podrá programar el número de repeticiones mediante los bits 0-3.

Los ejemplos (ENTER400.C, ENTER600.C, ET13H400.C, ET13X400.C) del CD de la revista muestran la programación de este registro. Un ejemplo práctico se puede ver en el listado 6, que programa la duplicación de líneas en el modo de 320x200 (13h).

PROGRAMACIÓN DE LA PALETA

Cualquier modo *chained* o *unchained* al que se acceda desde el modo 13h estándar hereda su sistema DAC de tripletes RGB. Esto quiere decir que para cambiar la paleta en Modo X pueden ser utilizadas exactamente las mismas funciones que se usaban para 13h, como puedan serlo el uso de funciones de la BIOS (interrupción 10h) o el sistema de acceso a registros DAC controlados por los puertos 3c7h, 3c8h y 3c9h, siendo este último sistema el más recomendable por su velocidad, como pudo verse en el **Curso de Programación de Demos** correspondiente al número 23 de Sólo Programadores.

PROYECTOS PARA EL LECTOR

Con lo aprendido en este curso de Modo X pueden hacerse desde verdaderas maravillas gráficas hasta utilidades con un espectacular entorno gráfico (imagine el lector, por, ejemplo una utilidad de copiado de discos en 480x300), pasando por juegos que probablemente aprovecharán el sistema de *scroll* de Modo X dibujando directamente en pantalla y restaurando el fondo al siguiente *frame*.

Sin olvidar el aprovechar la duplicación de líneas VGA para realizar efectos tan rápidos como el fuego de SP-BBS del directorio INFO que viene en el CD de la revista, que funciona a 70 frames por segundo en un 486 a 25 Mhz. Es posible así acelerar plasmas, fuegos, etc... que estén dibujados con ampliación 2x2, 4x4, etc...

MÁS SOBRE EL MODO X

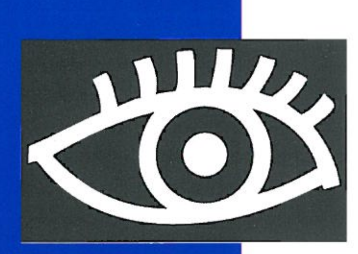
Para cualquier persona que quiera introducirse en conversaciones sobre el Modo X con expertos sobre el tema puede mirar en el *Mode X Frequent Answered Questions* (Mode X FAQ), situado en *comp.sys.ibm.pc.demos y rec.games.programmer*, donde podrá encontrar textos y programas del creador del Modo X, Michael Abrash, y de otros expertos en la materia.

Respecto a programas o utilidades sobre el modo X y los registros, el autor se permite recomendar el programa TWEAK de Robert Schmidt, una gran utilidad de lectura y modificación de registros de la VGA, así como todos los textos que circulan por Internet y BBSs acerca de este tema. Sobre estos textos, se han incluido algunos de ellos en el subdirectorio DOCS del software que acompaña al artículo. Estos ficheros son programas o listados que pueden ayudar a comprender diferentes aspectos de la programación en 13h para posteriormente ser adaptados a modo X según las necesidades de cada programador.

Las dudas sobre el tema pueden ser enviadas tanto a la revista como a Compiler Software. Ante todo, gracias por haber seguido atentamente este curso de programación avanzada de la sección indocumentada de la VGA.

PROGRAMACIÓN DEL GDI UTILIZANDO LAS MFC 4.0

Jorge R. Regidor



La programación gráfica para Windows pasa por la utilización de la librería de enlace dinámico GDI (*Graphics Device Interface*). Esta librería ofrece un conjunto de funciones, estructuras y mensajes destinados exclusivamente a facilitar dicha programación gráfica.

Windows es un entorno gráfico de trabajo, lo que se traduce que cualquier tipo de operación en pantalla, incluso escribir un texto, sea una operación gráfica. Por ello, el API de Windows ofrece una interfaz de programación para las operaciones gráficas, el cual es independiente de la resolución, número de colores, tipo de tarjeta, etc... Lo que permite que en las aplicaciones no haya que preocuparse por conocer qué tipo de tarjeta gráfica va a utilizar el usuario final de los programas. Sobre esta afirmación anterior se puede o se debe hacer matices, ya que por ejemplo, si se muestra una imagen o bitmap de 256 colores sobre una tarjeta que sólo soporta 16 el resultado puede ser realmente catastrófico. Aún en estos casos, se pueden evitar estas situaciones, ya que en tiempo de ejecución se puede averiguar las características de un dispositivo gráfico, y en consecuencia mostrar una imagen u otra, en función de los colores soportados. Por estos motivos y por otros similares, es bueno tener un conocimiento amplio de los contextos de dispositivo, que informan sobre las capacidades gráficas de dicho dispositivo.

CONTEXTOS DE DISPOSITIVO Y OBJETOS GRÁFICOS

Un contexto de dispositivo (*Device Context*) es una estructura que define todos los objetos, atributos y los modos que definen un dispositivo gráfico. Estos dispositivos pueden ser desde la pantalla hasta una impresora, pasando incluso por los *metafiles*.

Después de esta definición habría que definir de igual modo qué es un objeto gráfico. Los objetos gráficos son herramientas de dibujo suministradas por el API de Windows para el programador, al igual que las puede ofrecer un programa de diseño gráfico al diseñador (Ver Figura A). Dentro de los objetos gráficos se pueden encontrar las plumas (*Pens*) que permiten dibujar líneas de diferentes grosores, las brochas (*Brushes*) que permiten rellenar fondos y figuras, las imágenes (*Bitmaps*) que permiten mostrar imágenes, las paletas (*Palette*) que permiten definir la paleta de colores a utilizar en un contexto de dispositivo, las fuentes (*Fonts*) que permiten mostrar texto con diferentes tipografías y tamaños y las regiones (*Region*) que permiten definir regiones para ser pintadas, rellenadas o detectar pulsaciones sobre ellas. Cada contexto de dispositivo tiene asociados una serie de estos objetos gráficos, que junto con sus propios estados definen el estado general del contexto de dispositivo.

Además, se ha hecho referencia a los *metafiles*, que permiten guardar en un fichero una serie de operaciones gráficas para luego poder ser reproducidas. De este modo en vez de almacenar el resultado de una imagen se almacenan las operaciones de forma ordenada, las cuales normalmente ocupan menos espacio que el resultado final, aunque también son más lentas a la hora de cargarse.

GDI Y VISUAL C++ 4.0

La programación gráfica para Windows pasa por la utilización de la librería de

enlace dinámico GDI (*Graphics Device Interface*). Esta librería ofrece un conjunto de funciones, estructuras y mensajes destinados exclusivamente a facilitar dicha programación gráfica. Visual C++ y su librería de clases MFC 4.0 contienen varias clases que facilitan la programación gráfica en el entorno Windows, encapsulando la librería GDI bajo varias clases de fácil utilización.

Las clases destinadas para la programación gráfica en las MFCs se pueden dividir en dos grupos, las encargadas de controlar los contextos de dispositivo, derivadas de la clase *CDC*, (ver Figura B) y las encargadas de controlar los objetos gráficos existentes en Windows, derivadas de *CGdiObject* (ver Figura C).

La clase encargada de manejar los contextos de dispositivos es *CDC*, de la cual se derivan las clases *CClientDC*, *CWindowDC*, *CPaintDC* y *CMetaFileDC*.

LA CLASE CDC

CDC es la clase base de todas las que permiten utilizar los contextos de dispositivo. Si es usada directamente, permite acceder a toda la pantalla, así como a otros dispositivos gráficos como impresoras.

Hay muchas funciones miembro de esta clase que permiten realizar operaciones con los contextos de dispositivo, utilizar los diferentes objetos gráficos, selección segura de objetos gráficos y trabajo con paletas y colores.

La clase *CDC* contiene dos contextos de dispositivo, referenciados por *m_hDC* y *m_hAttribDC*, que en un principio coinciden, pero que en algunos casos pueden ser diferentes y utilizados *m_hDC* como contexto de salida y *m_AttribDC* como contexto para coger atributos y estados.

Las diferentes funciones miembro de la clase *CDC* se pueden clasificar en los siguientes grupos principales, de los cuales se muestran algunas funciones:

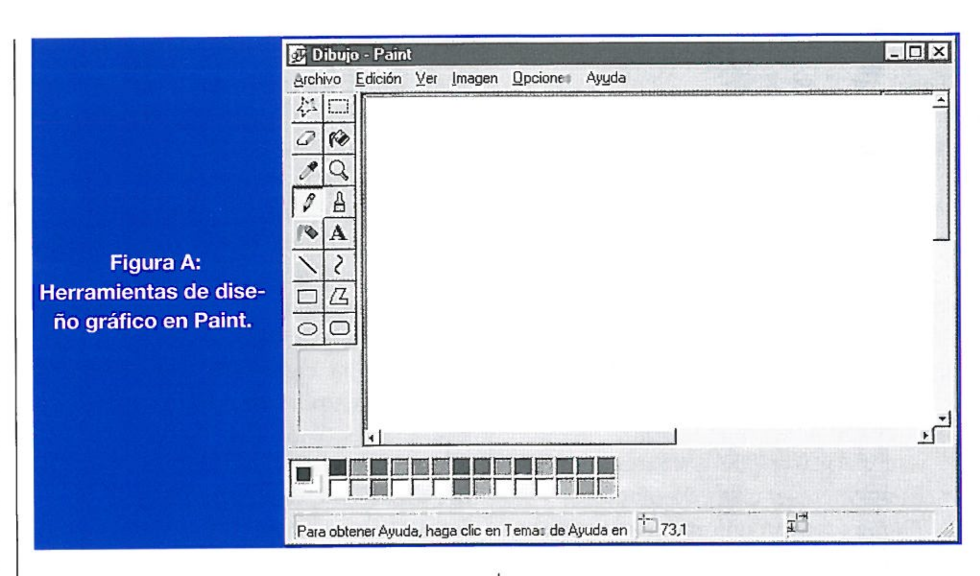


Figura A: Herramientas de diseño gráfico en Paint.

Windows es un entorno gráfico de trabajo, lo que quiere decir que cualquier tipo de operación en pantalla será una operación gráfica

- Construcción/Destrucción.

  CDC() construye un objeto de esta clase.
- Inicialización.

  CreateDC crea un contexto para un determinado dispositivo.

  CreateIC crea un contexto de información para un determinado dispositivo.

  CreateCompatibleDC crea un contexto en memoria compatible con otro contexto de un dispositivo.

  GetCurrentBitmap devuelve un puntero del tipo *CBitmap* al *bitmap* actualmente seleccionado.

  GetCurrentBrush devuelve un puntero del tipo *CBrush* a la brocha actualmente seleccionada.

  GetCurrentPen devuelve un puntero del tipo *CPen* al pincel actualmente seleccionado.

  GetCurrentFont devuelve un puntero del tipo *CFont* a la fuente actualmente seleccionada.

  GetCurrentPalette devuelve un puntero del tipo *CPalette* a la paleta actualmente seleccionada.
- Funciones de contexto de dispositivo.

  GetSafeHdc devuelve el *handle* al contexto de dispositivo apuntado por *m_hDC*.

  SaveDC graba el estado actual del contexto de dispositivo.

  RestoreDC restaura el último estado grabado del contexto de dispositivo.

  GetDeviceCaps devuelve las capacidades de un determinado dispositivo. Esta función devuelve un determinado tipo de información en función del índice que se le indique. Los índices pueden verse en la tabla 1.
- Funciones de herramientas de dibujo.

  GetBrushOrg devuelve el origen actual de la brocha.

  SetBrushOrg establece el origen actual de la brocha.

  EnumObjects enumera los pinceles y las brochas disponibles en el contexto de dispositivo.
- Selección segura de objetos gráficos.

  SelectObject selecciona un objeto del GDI en el contexto de dispositivo.

  SelectStockObject selecciona un objeto predefinido del GDI en el contexto de dispositivo.
- Funciones para color y paletas de colores.

  GetNearestColor devuelve el color sólido más cercano a un color RGB indicado.

  SelectPalette selecciona una paleta lógica de colores.

  RealizePalette cambia la paleta lógica por la paleta del sistema.
- Funciones para modificar los atributos de dibujo.

  GetBkColor devuelve el color de fondo del contexto.

  GetBkMode devuelve el modo de fondo del contexto.

  GetTextColor devuelve el color del texto del contexto.

SetBkColor establece el color de fondo del contexto.

SetBkMode establece el modo de fondo del contexto.

SetTextColor establece el color del texto del contexto.

- Funciones para líneas.

GetCurrentPosition devuelve la posición actual del pincel.

MoveTo mueve el pincel a una determinada posición.

LineTo pinta una línea desde la posición actual hasta la indicada.

Arc y *ArcTo* pintan un arco.

- Funciones sencillas de dibujo.

FillRect rellena un rectángulo usando una brocha.

DrawIcon dibuja un icono.

- Funciones para elipses y polígonos.

Ellipse dibuja una elipse.

Pie dibuja un gráfico de tarta.

Rectangle dibuja un rectángulo.

- Funciones para bitmaps.

BitBlt copia un bitmap desde un contexto de dispositivo a otro.

GetPixel devuelve el color de un pixel indicado.

SetPixel establece el color de un pixel indicado.

- Funciones para texto.

TextOut pinta un texto en la posición indicada y con la fuente seleccionada.

DrawText pinta un texto con formato en el rectángulo indicado.

- Funciones de *scroll*.

ScrollDC realiza un *scroll* tanto vertical como horizontal del contexto de dispositivo.

- Funciones para *metafiles*.

PlayMetaFile carga un *metafile* y lo copia sobre el contexto de dispositivo.

LA CLASE CCLIENTDC

Esta clase permite utilizar el contexto de dispositivo de la pantalla que se corresponde con el área de cliente de una ventana. El área de cliente, como bien sabemos, se corresponde con la zona de la ventana donde se muestra la salida de la aplicación, ya sea texto o gráficos. Esta clase es normalmente utilizada como respuesta a la detección de algún tipo de evento, normalmente del ratón.

La clase *CClientDC* llama a la función *GetDC* al llamar a su constructor, y a la función *ReleaseDC* al llamar al destructor, liberando al programador de esta labor. Esta clase se deriva directamente de la clase *CDC*, añadiendo la variable miembro *m_hWnd*, que identifica a la ventana.

LA CLASE CWINDOWDC

Esta clase permite utilizar el contexto de dispositivo de la pantalla que se corresponde con el área total de una ventana. Esta zona corresponde al área de cliente más las zonas, como los bordes y la barra de título.

La clase *CWindowDC* llama a la función *GetWindowDC* al llamar a su constructor y a la función *ReleaseDC* al llamar al destructor, liberando al programador de esta labor. Esta clase se deriva directamente de la clase *CDC*, añadiendo la variable miembro *m_hWnd*, que identifica a la ventana.

Los objetos gráficos son herramientas de dibujo suministradas por el API de Windows para el programador

LA CLASE CPAINTDC

La clase *CPaintDC* debe ser utilizada exclusivamente en la función *OnPaint*, que se corresponde con el mensaje *WM_PAINT*, y se encarga de llamar a la función *BeginPaint* en el constructor y a la función *EndPaint* en el destructor, por lo que no habrá que llamar a dichas funciones. Esta clase se deriva directamente de la clase *CDC*, añadiendo las variables miembro *m_hWnd*, que identifica a la ventana y *m_ps*, que contiene una estructura *PAINTSTRUCT* y que es llenada al llamar a la función *CWnd::BeginPaint*.

LA CLASE CMETAFILEDC

Esta clase es utilizada para grabar operaciones gráficas sobre los *metafiles* de Windows. Para trabajar con estos contextos se debe crear un objeto con el constructor *CMetaFileDC*. Posteriormente se debe llamar a la función *Create* para crear un contexto de dispositivo y asociarlo con el objeto. Después de esto, se pueden llamar a las funciones que produzcan salida sobre el contexto, las cuales serán almacenadas sobre el *metafile* al llamar a la función *Close*.

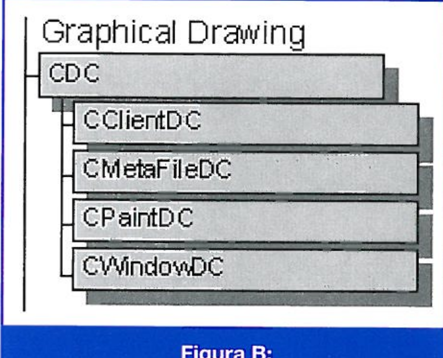


Figura B: Jerarquía de clases para contextos de dispositivo.

OBJETOS DEL GDI

Como ya se ha indicado, Windows ofrece una serie de objetos o herramientas que permiten dibujar líneas con los pinceles, rellenar figuras con las brochas, mostrar imágenes. Los pasos necesarios para utilizar correctamente los objetos gráficos dentro de los contextos de dispositivo son:

1- Crear un objeto de la clase necesaria, por ejemplo, de la clase *CPen*, y asociar a éste objeto un pincel con la función miembro *CreatePen*.
2- Seleccionar el objeto gráfico dentro del contexto de dispositivo y guardar el objeto gráfico anterior.
3- Después de haber utilizado el objeto gráfico, reponer el antiguo.
4- Eliminar el objeto utilizado si fuese necesario.

Además de estos pasos hay que tomar algunas precauciones, como por ejemplo el objeto gráfico devuelto al seleccionar el que se quiere utilizar es temporal, y por tanto su uso no debe salir de la función donde se trate. Otra precaución a tomar está relacionada con la forma en la que se crea el objeto gráfico. Existen dos métodos, uno



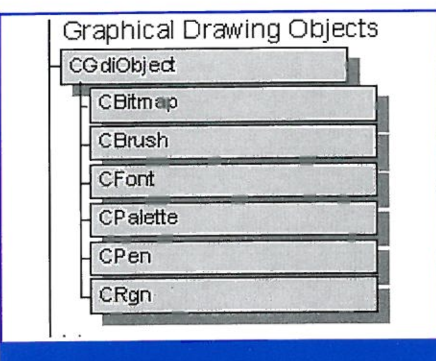


Figura C:
Jerarquía de clases para objetos gráficos.

basado en el constructor y otro llamando a la función miembro *CreateXXX*. Si se utiliza el primer método no se puede comprobar si ha existido un error o no, pero si se llama a la función *Create* se puede comprobar su buen funcionamiento con el resultado de la función. En las siguientes líneas se muestra el método recomendado para la creación de objetos gráficos.

```
CPen pen;
if( pen.CreatePen( PS_SOLID, 2,
RGB(0,0,0) ) )
{
  // Utilizar el objeto
}
else
{
  // Indicar el error
}
```

LA CLASE CPEN

La clase *CPen* encapsula las operaciones sobre el objeto gráfico de Windows que permite dibujar líneas. Esta clase tiene varios constructores, que permiten crear objetos sin asociarle un objeto gráfico o crearlo y asociarlo directamente.

Los pinceles en Windows pueden ser de los siguientes tipos:

- *PS_SOLID*: Un pincel sólido.
- *PS_DASH*: Crea un pincel de líneas discontinuas.
- *PS_DOT*: Crea un pincel de puntos.
- *PS_DASHDOT*: Crea un pincel con un punto y una línea.
- *PS_DASHDOTDOT*: Crea un pincel con dos puntos y una línea.
- *PS_NULL*: Crea un pincel nulo.
- *PS_INSIDEFRAME*: Crea un pincel que puede ser rellenado con una brocha, como los rectángulos o las elipses.

Existe una segunda versión de *CPen* donde se le puede indicar tipo, estilo, final y atributos para uniones. Los tipos de pinceles pueden ser *PS_GEOMETRIC* y *PS_COSMETIC*. Los estilos pueden ser *PS_ALTERNATE* y *PS_USERSTYLE*. Los finales pueden ser *PS_ENDCAP_ROUND*, *PS_ENDCAP_SQUARE* y *PS_ENDCAP_FLAT*. Por último, los atributos de unión pueden ser *PS_JOIN_BEVEL*, *PS_JOIN_MITER* y *PS_JOIN_ROUND*.

Las operaciones que sobre la clase *CPen* se pueden realizar son obtener el *handle* del objeto gráfico con el operador *HPEN* y obtener la estructura que contiene la información de dicho objeto, ya sea de la versión normal mediante *GetLogPen* o la versión extendida con *GetExtLogPen*. Además, se puede obtener el puntero del tipo *CPen* dándole un *handle* con la función *FromHandle*.

Dentro de las funciones de los dispositivos de contexto hay que resaltar que aquellas que utilizan los pinceles son *Line*, *LineTo*, *Arc*, *ArcTo*, *AngleArc*, *PolyLine*, *PolyBezier* y todas aquellas que dibujen líneas.

LA CLASE CBRUSH

Esta clase encapsula todas las operaciones que se pueden realizar para crear y modificar las brochas de Windows. Del mismo modo que la clase anterior, el constructor permite asociar o no el objeto gráfico. Si se asocia el objeto gráfico a posteriori, se pueden utilizar cualquiera de las siguientes funciones:

- *CreateSolidBrush*: Inicializa la brocha asociándola un color sólido.
- *CreateHatchBrush*: Inicializa la brocha asociándola un patrón de líneas con formatos como *HS_BDIAGONAL*, que crea líneas descendentes a 45 grados de izquierda a derecha; *HS_CROSS*, que crea líneas cruzadas; *HS_DIAGCROSS*, que crea líneas cruzadas a 45 grados; *HS_FDIAGONAL*, que crea líneas ascendentes a 45 grados de izquierda a derecha; *HS_HORIZONTAL*, que crea líneas horizontales y *HS_VERTICAL*, que crea líneas verticales.
- *CreatePatternBrush*: Inicializa la brocha con un *bitmap* especificado.
- *CreateDIBPatternBrush*: Inicializa la brocha con un *bitmap* independiente de dispositivo especificado.
- *CreateSysColorBrush*: Inicializa la brocha con uno de los colores del sistema.

Las operaciones que se pueden realizar sobre los objetos de la clase *CBrush* son obtener el *handle* del objeto gráfico con el operador *HBRUSH*, obtener la estructura que contiene toda la información de la brocha con *GetLogBrush* y obtener un puntero al objeto *CBrush* a partir del *handle* con la función *FormHandle*.

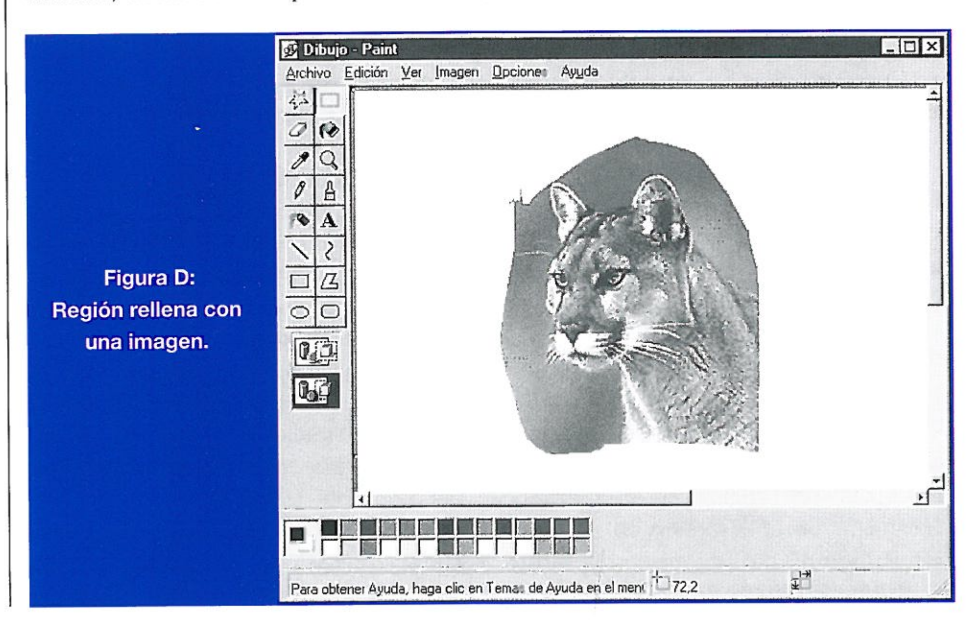


Figura D:
Región rellena con una imagen.

Las brochas son utilizadas por funciones de los contextos de dispositivo como *Rectangle*, *Pie*, *Ellipse*, *FillRect* y todas aquellas en las cuales haya que rellenar una zona o figura.

LA CLASE CFONT

La clase *CFont* encapsula el acceso a las fuentes utilizadas por Windows y ofrece funciones para manipularlas. Las programación de fuentes es un tema amplio, y con muchos conceptos relacionados con la tipografía.

El único constructor de la clase *CFont* no permite asociar una fuente directamente, por lo que hay que llamar a funciones como *CreateFont* Indirecto o *CreateFont* después de haber creado el objeto.

Las operaciones que esta clase permite son obtener el *handle* de la fuente con el operador *HFONT*, obtener la estructura que contiene toda la información de la fuente mediante *GetLogFont* y obtener un puntero a un objeto *CFont* a partir de su *handle*.

LA CLASE CBITMAP

La clase *CBitmap* encapsula los más potentes y complejos objetos gráficos como son los *bitmaps*, mapas de bits o simplemente imágenes. Esta clase, al igual que *CFont*, sólo tiene un constructor, que además nó asocia ningún *bitmap*. Para esto hay que llamar a alguna de las funciones que asocian la imagen, como son:

- *LoadBitmap*: Asocia una imagen desde el fichero de recursos de la aplicación.
- *LoadOEMBitmap*: Asocia una imagen predefinida por Windows. Los posibles valores pueden verse en la ayuda en línea de la función *CBitmap::LoadOEMBitmap* o en el fichero *Windows.h*.
- *CreateBitmap*: Crea un *bitmap* a partir de los datos suministrados. Este método suele ser el habitual si se carga desde un fichero BMP.

Las operaciones que se pueden realizar sobre esta clase son obtener el *handle* del *bitmap* con el operador *HBITMAP*, obtener la estructura con la información del *bitmap* mediante *GetBitmap*, obtener un puntero a un objeto *CBitmap* desde su *handle*, obtener y establecer los bits que definen el *bitmap* con las funciones *GetBitmapBits* y *SetBitmapBits*, y obtener y establecer las dimensiones del mismo con las funciones *GetBitmapDimension* y *SetBitmapDimension* respectivamente.

Las funciones del contexto de dispositivo que utilizan los *bitmaps* son *BitBlt*, *StretchBlt*, *PatBlt*, *GetPixel*, *SetPixel*, *FloodFill* y todas las demás funciones que sirvan para modificar u operar con o sobre *bitmaps*.

Índice	Significado
DRIVERVERSION	Versión del dispositivo.
TECHNOLOGY	Tecnología del dispositivo.
HORZSIZE	Tamaño horizontal, en milímetros.
VERTSIZE	Tamaño vertical, en milímetros.
HORZRES	Resolución horizontal, en pixeles.
VERTRES	Resolución vertical, en pixeles.
LOGPIXELY	Número de pixeles por pulgada en altura.
LOGPIXELX	Número de pixeles por pulgada en anchura.
BITSPIXEL	Número de bits por pixel.
PLANES	Número de planos de color.
NUMBRUSHES	Número de brochas del dispositivo.
NUMPENS	Número de pinceles del dispositivo.
NUMFONTS	Número de fuentes del dispositivo.
NUMCOLORS	Número de colores del dispositivo.

Figura D:
Región rellena con una imagen.

La clase CPaintDC debe ser utilizada exclusivamente en la función OnPaint, que se corresponde con el mensaje WM_PAINT

LA CLASE CPALETTE

Una paleta permite definir los colores que va a utilizar una aplicación de un determinado dispositivo de salida en color, como por ejemplo una tarjeta VGA. Estos objetos gráficos permiten obtener las máximas capacidades de color de un dispositivo sin afectar de modo dramático a las demás aplicaciones. Para ello, cada aplicación tiene una paleta lógica, que se carga y descarga al activarse o desactivarse cada aplicación. La clase *CPalette* encapsula el acceso a este objeto gráfico ofreciendo funciones para la creación y manipulación de las paletas. La clase *CPalette* sólo tiene un constructor que no asocia ninguna paleta al objeto. Para ello habrá que llamar a las funciones *CreatePalette* y *CreateHalftonePalette*.

Las operaciones sobre esta clase permiten obtener el *handle* de la paleta con el operador *HPALETTE*, obtener el número de entradas (colores) que contiene la paleta con *GetEntryCount*, obtener y establecer un rango de entradas con *GetPaletteEntries* y *SetPaletteEntries* respectivamente, obtener el índice de entrada del color que más se aproxime a uno dado con *GetNearestPaletteIndex*, obtener el puntero del tipo *CPalette* desde su *handle* con *FromHandle*, cambiar el tamaño (número de entradas) de la paleta con *ResizePalette* y sustituir un rango de entradas por uno dado y actualizar los colores de la aplicación en una sola operación con la función *AnimatePalette*.

Las funciones del contexto de dispositivos que hacen uso de la clase *CPalette* son *SelectPalette*, que selecciona una paleta como paleta del con-

texto y *RealizePalette*, que permite establecer la paleta como activa en una aplicación.

LA CLASE CRGN

Una región es una área elíptica o poligonal que puede ser rellenada y permite enviar eventos como pulsaciones del ratón sobre el área. La clase *CRgn* permite crear, modificar y obtener información sobre la región a la que pertenece. La clase *CRgn* sólo tiene un constructor que no asocia ninguna región, por lo que se debe llamar a las funciones siguientes para asociar una región al objeto.

- *CreateRectRgn*: Crea una región rectangular.
- *CreateEllipticRgn*: Crea una región elíptica.
- *CreatePolygonRgn*: Crea una región poligonal.
- *CreatePolyPolygonRgn*: Crea una región que consiste en una serie de polígonos cerrados.
- *CreateRoundRectRgn*: Crea una región rectangular con los bordes redondeados.
- *CombineRgn*: Combina dos regiones creando una tercera.

Windows ofrece una serie de objetos o herramientas que permiten dibujar líneas con los pinceles, rellenar figuras con las brochas o mostrar imágenes

Con esta clase se pueden comparar dos regiones para saber si son equivalentes con la función *EqualRgn*, obtener el puntero al objeto *CRgn* desde el *handle* de la región con la función *FromHandle*, obtener el rectángulo externo a la región mediante *GetRngBox*, mover la región con *OffsetRgn*, ajustar una región a un rectángulo con *SetRectRgn*, determinar si un punto pertenece o no a una región con *PtInRegion*, determinar si una región está contenida en un rectángulo y también obtener el *handle* de la región con el operador *HRGN*.

Las funciones de los contextos de dispositivos asociados con las regiones son *FillRgn*, que permite rellenar una región con una brocha; *FrameRgn*, que permite dibujar un borde usando de nuevo una brocha; *InvertRgn*, que invierte el color de una región y *PaintRgn*, que rellena una región con la brocha seleccionada en el contexto de dispositivo.

Figura D: Región rellena con una imagen.

CONCLUSIÓN

En este artículo se ha pretendido dar un repaso general a los objetos que componen la librería GDI. Se ha tratado de dar una visión global para tener consciencia de las operaciones que se pueden realizar y de la potencia ofrecida por todas las clases, tanto las encargadas de encapsular los contextos de dispositivos como las encargadas de encapsular los diferentes objetos gráficos que componen el mundo gráfico en Windows. En las próximas entregas se entrará en detalle y se verán los métodos de programación de los diferentes objetos gráficos dentro de los contextos de dispositivo.

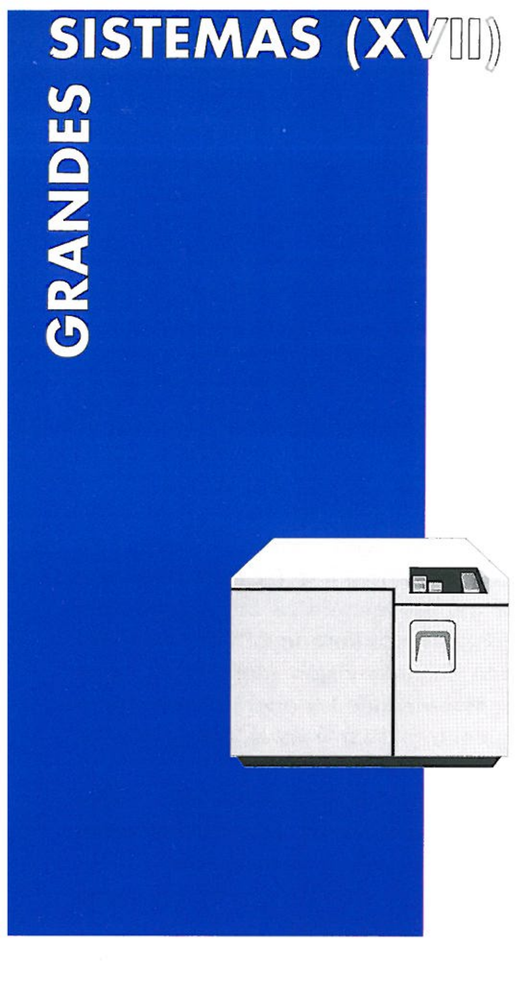
CONCLUSIÓN

En este articulo se ha pretendido dar un repaso general a los objetos que componen la librería GDI. Se ha preferido dar una visión global para tener consciencia de las operaciones que se pueden realizar y de la potencia ofrecida por todas las clases, tanto las encargadas de encapsular los contextos de dispositivos, como las encargadas de encapsular los diferentes objetos gráficos que componen el mundo gráfico en Windows.

Básicamente, y como adelanto al siguiente articulo, podemos decir que el modo de utilización de estos objetos gráficos, pasa por las siguientes fases:

1. Crear el objeto gráfico con los parámetros deseados.
2. Seleccionar el objeto dentro del contexto de dispositivo de la ventana, y guardar el objeto gráfico anterior.
3. Realizar las operaciones gráficas que utilicen el objeto gráfico seleccionado.
4. Reestablecer el objeto gráfico anterior, para no afectar a otras aplicaciones.
5. Destruir el objeto gráfico ya utilizado.

El modo de realizar estas operaciones sobre algunos de los objetos gráficos anteriormente vistos, lo veremos en el siguiente articulo, donde crearemos una aplicación que realice diversas operaciones sobre pantalla.



UTILIDADES MVS (II)

José María Peco

Este mes se continúa con el tema dedicado a las Utilidades que acompañan al Sistema Operativo MVS de IBM. Estas utilidades son herramientas que por su sencillez y potencia permiten realizar gran número de funciones a nivel del sistema. Son las típicas utilidades del sistema operativo.

El pasado mes se inició un tema de utilidad en una gran instalación, tal y como puede comprobar cualquier desarrollador de este entorno a diario. Las utilidades que se trataron en ese primer artículo, fueron:
IEFBR14 para inicializar o borrar ficheros.
IEBGENER para copiar ficheros y manipular su contenido.
IEHLIST para listar nombres de ficheros o miembros.

El autor se ha permitido clasificar estas utilidades como "utilidades de la primera época", ya que acompañan al sistema operativo desde sus primeras versiones, incluso con su predecesor, el DOS/VS.

No es intención del autor trasladar a este artículo el manual de referencia de estas utilidades, pero la experiencia demuestra que la barrera que existe en estas instalaciones entre el entorno de sistemas y el entorno de desarrollo impide el acceso físico a dichos manuales hace que se den casos tan paradójicos como el que se le presentó días antes de escribir esta entrega, cuando cayó en sus manos un JCL que "para borrar un fichero" invocaba a NATURAL, y una vez en este entorno, haciendo uso de las facilidades del producto NATURAL-PROCESS borraba dicho fichero, con lo cual lo que se está haciendo es elevar el consumo de recursos a límites insospechados por tener que alocar, al menos, todos los ficheros del entorno NATURAL, y todo ello por no conocer la utilidad IEFBR14 vista el pasado mes.

Particularmente se recomienda insistir en todas aquellas instalaciones en las que no se disponga de los manuales de IBM, para conseguir una copia de los mismos, ya que copiar las distintas utilidades desde otro JCL que haga algo parecido no resuelve el problema, pues obligará probablemente a dar más pasos de los necesarios o bien, dependiendo de la iniciativa del programador, a utilizar los medios del entorno que éste conozca, como en el ejemplo comentado anteriormente, aunque esto sea más costoso en recursos del sistema.

REGLAS DE LAS FICHAS SYSIN

Los comandos que debe ejecutar la utilidad se agrupan en la *Ddname* SYSIN, la cual suele ir *In-stream* (es decir, en la propia cadena), aunque puede perfectamente especificarse dentro de cualquier otro fichero, asociando su *Ddname* a ésta.

A modo de recordatorio, se citan las reglas que deben seguirse para especificar estos comandos:

- El nombre del comando tiene que ser un nombre válido para la utilidad.
- Los operandos van separados del comando por, al menos, un blanco.
- Los distintos operandos de un comando van concatenados entre sí, sirviendo la coma (,) como delimitador de los mismos.
- El tercer grupo de elementos de una ficha de control son los comentarios, que se encuentran separados de los operandos por, al menos, un blanco.
- La codificación de las fichas de control no puede sobrepasar la columna 71 (Ojo con la numeración propia del fichero que contiene el JCL, que puede provocar un error en el caso de que tenga definido *NUM STD ON*)
- Cuando no cabe la codificación de una ficha en una línea, se detiene la codificación en la columna 71 o después de una coma, y se escribe algo (distinto de blanco) en la columna

72, continuando esta codificación en la siguiente línea a partir de la columna 16.

CARACTERES 'ASA'

El control de carro para la impresión de ficheros se realiza incorporando estos caracteres ASA en la primera columna de aquellos ficheros que tengan definido en la DCB (*Description Control Block*) el formato de registro (Parámetro *RECFM*) FBA o VBA (Fijo/Variable Bloquado ASA)

Los caracteres ASA con significado son los que se muestran en la figura 1.

Figura 1

```
B    Imprime la linea despues de un salto de carro
0    Imprime la linea despues de dos saltos de carro
-    Se imprime la linea despues de tres saltos de carro
N    Se imprime la linea sin saltar el carro (sobreimprime)
+N   Se sobreimprime la linea N veces sin saltar el carro.
1    Se salta de Pagina.
*    La linea no se imprime por tratarse como comentario.
```

IEHPROG

Esta utilidad permite:

- Borrar, catalogar, descatalogar y renombrar un fichero.
- Borrar y renombrar un miembro de un fichero particionado.

Los comandos admitidos por esta utilidad a través de la SYSIN son:

- *CATLG* : Cataloga un fichero.
- *UNCATLG* :Descataloga un fichero.
- *SCRATCH* :Borrar un fichero.
- *RENAME* :Renombra un fichero.

Como ya se ha repetido en varias ocasiones, un fichero se encuentra catalogado cuando su nombre figura en un fichero de la instalación que contiene todos los nombres de los ficheros de la instalación y, para cada uno de ellos, el nombre del volumen (o disco físico) en el que se encuentra físicamente. Por lo tanto, para catalogar un fichero, sólo hay que especificar dos parámetros: DSN (*Data_Set_Name*) o nombre del fichero, y el nombre del volumen en el que se encuentra.

Los comandos que debe ejecutar la utilidad se agrupan en la Ddname SYSIN

Los siguientes ejemplos muestran como catalogar dos ficheros, uno en disco (SYSDA) y otro en cinta (TAPE):

CATLG DSN=JMPDES.FICHEROD,VOL =SYSDA=VOLUM11
CATLG DSN=JMPDES.FICHEROC,VOL =TAPE=(002365,1,000153,1)

Por el contrario, para descatalogar un fichero sólo hay que borrarle del catálogo. Por lo tanto, sólo hay que especificar el nombre del fichero a descatalogar. Nótese que sólo se borra la entrada del catálogo, pero no se borra el fichero, el cual seguirá residiendo en el disco o cinta en el que se encuentre. Ejemplo:

UNCATLG DSN=JMPDES.FICHEROD

Para borrar un fichero físicamente se debe utilizar el comando *SCRATCH*. Este comando accede directamente a la VTOC (*Volume Table Of Contains* - Tabla de Contenido del Volumen) del volumen que se especifica como parámetro, y borra la entrada correspondiente, liberando el espacio usado. Ejemplo:

SCRATCH DSN=JMPDES.FICHEROD, VOL=SYSDA=VOLUM11

Así mismo, se recuerda que los ficheros particionados están formados por dos partes: el área de directorio, que contienen los nombres o índice de los miembros, y el área de datos, que contiene los datos, o contenido propio de los distintos miembros. Por lo tanto, cuando se desea borrar un miembro de un PDS (*Particioned Data Set*) se debe especificar el nombre del fichero y el volumen en el que se encuentra para poder localizar el área de directorio en la VTOC. Y una vez localizado, borrar de dicho área la entrada correspondiente al miembro a borrar, pero no se libera el espacio hasta que se comprima la librería (Sólo Programadores núm. 19).

Ejemplo:

SCRATCH DSN=JMPDES.FICHEROP, VOL=SYSDA=VOLUM11,MEMBER=EJE MPLO1

Por ultimo, para renombrar ficheros, se usa el comando *RENAME* y los parámetros DSN para especificar el nombre actual, y *NEWNAME* para especificar el nombre nuevo.

Cuando se trata de usar este comando para renombrar miembros de un PDS, se debe especificar el nombre del miembro y el parámetro *NEWNAME*:

RENAME DSN=JMPDES.FICHERO1, VOL=SYSDA=VOLUM11,NEWNAME=J MPDES.FICHERO9
RENAME DSN=JMPDES.FICEHROP, VOL=SYSDA=VOLUM11, MEMBER=EJEMPLO1,NEWNAME=EJE MPLO9

Esta utilidad también tiene como particularidad la citada el pasado mes con la utilidad IEHLIST, en el sentido de que éste programa, para poder ejecutarse, debe contener fichas DD para alocar (incorporar al entorno) el volumen que se referenciará en el comando a ejecutar en la SYSIN, tal y como muestra el ejemplo de la figura 2.

IEBPTPCH

Este programa de utilidad se usa para imprimir o perforar en ficha o cinta de teletipo:

- Un fichero secuencial o particionado completo.
- Miembros específicos de un fichero particionado
- Registros seleccionados de un fichero secuencial o particionado.
- El directorio de un fichero particionado.
- Versión editada de un fichero secuencial o particionado.

El nombre del fichero de entrada se especifica asociándole a la *Ddname*

SYSUT1, y el de salida a la SYSUT2, que normalmente coincidirá con una cola de salida del *spool* la cual, por defecto, tiene la siguiente DCB:

DCB=(LRECL=121,BLKSIZE=121,RECFM=FBA).

Los comandos admitidos por esta utilidad a través de la Ddname SYSIN son:

- *PRINT* Imprime un fichero.
- *RECORD* Especifica el formato del registro de salida.
- *MEMBER* Especifica el nombre del miembro a listar.
- *TITLE* Especifica el literal de la cabeceras de las paginas.

Los operadores del comando PRINT son:

TYPORG=PO para Organización Particionada y *PS* para organización Secuencial.

MAXFLDS=n, donde *n* especifica el numero máximo de parámetros *FIELD* que se especificarán con el comando *RECORD*.

MAXNAME=m, donde *m* especifica el numero máximo de parámetros *NAME* que se especificarán con el comando *MEMBER*.

STOPAFT=nn, donde *nn* especifica el numero de registros que se listarán de cada miembro.

SKIP=ss, donde *ss* especifica que imprima una línea en blanco cada *ss* registros impresos.

STRTAFT=pp, donde *pp* especifica que empiece a imprimir ignorando los *pp* registros iniciales.

Los operadores del resto de comandos se muestran junto con un ejemplo:

a) Imprimir los miembros que se especifican:

MEMBER NAME=miembro1,NAME=miembro2

b) Imprimir los primeros 80 bytes del registro de entrada:

RECORD FIELD=(80)

c) Imprimir 5 asteriscos en las primeras posiciones, y en las posiciones 6 y siguientes los 10 bytes que se encuentran en las posiciones 71 y siguientes del registro de entrada:

*RECORD FIELD=(5,'*****',,1),FIELD=(10,71,,6)*

d)Imprimir un literal como título dejando un *offset* de 40 posiciones, es decir, dejando 40 blancos a la izquierda:

TITLE ITEM=('CABECERA LISTADO', 40)

La figura 3 muestra un ejemplo de uso de esta utilidad, mediante la cual se desea poner un título al listado del fichero que contiene los nombres de los objetos a listar.

Figura 2

```
//EJEMPLO1  EXEC PGM=IEHPROG
//SYSPRINT  DD *
CATLG       DSN=JMPDES.FICHERO1,VOL=SYSDA=VOLUM11
UNCATLG     DSN=JMPDES.FICHERO2
SCRATCH     DSN=JMPDES.FICHERO3,VOL=SYSDA=VOLUM11,PURGE
RENAME      DSN=JMPDES.FICHERO4,VOL=SYSDA=VOLUM11,
            MEMBER=VIEJO,NEWNAME=NUEVO
/*
//VOLUMEN   DD UNIT=SYSDA,VOL=SER=VOLUM11,DISP=SHR
```

IEBCOPY

Esta *utility* es usada para:

- Copiar ficheros particionados a secuencial.
- Restaurar ficheros desde cinta a disco.

Si la DD de entrada está asociada a un fichero particionado, y la DD de salida está asociada a un fichero secuencial, copia todo el fichero PDS sobre el secuencial de salida.

La particularidad de esta utilidad se centra en el hecho de que los nombres de los ficheros de entrada y de salida no se referencian asociándoles a las *Ddnames* SYSUT1 o SYSUT2, sino que se referencian especificando *Ddnames* en las fichas DD que contienen los nombres de estos ficheros, y luego referenciando estas *Ddnames* en el comando, como se verá a continuación.

Los comandos admitidos por esta utilidad a través de la SYSIN son:

- *COPY*:Define las *Ddnames* asociadas a los ficheros de entrada (*INDD*) y de salida (*OUTDD*). El siguiente ejemplo concatena dos librerías de entrada:

COPY INDD=(DDENTRA,DDENTRA2), OUTDD=DDSALIDA

- *SELECT*: Define los nombres de los miembros a copiar mediante el parámetro *MEMBER=(miembro1,miembro2,...)*
- *EXCLUDE*: Especifica que se copien todos los miembros de la librería de entrada, excluyendo los miembros cuyos nombres se relacionan con el parámetro *MEMBER*:

El nombre del comando debe ser un nombre válido para la utilidad

```
//*==============================================================
//* IMPRIMIR FICHERO CON LOS NOMBRES DE LOS OBJETOS A LISTAR
//*==============================================================
//CABECERA EXEC PGM=IEBPTPCH
//STEPLIB  DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//*           --------- FICHERO DE ENTRADA --------------
//SYSUT1   DD *
**************************************XXXXXXXX
   LA ENTRADA ESTA FORMADA POR ESTAS LINEAS DE
   TEXTO Y POR EL FICHERO JMPDES.ENTRADA
**************************************XXXXXXXX
/*
//         DD DISP=SHR,DSN=JMPDES.ENTRADA
//*           --------- FICHERO DE SALIDA --------------
//SYSUT2   DD SYSOUT=*,HOLD=YES
//SYSIN    DD *
  PRINT TYPORG=PS,MAXFLDS=1
  RECORD FIELD=(40,1,,30)
  TITLE  ITEM=('DSN : JMPDES.ENTRADA',30)
  TITLE  ITEM=('*****',30)
/*
```

Figura 3

Clist para editar un JCL, cambiarle y submitir resultado

```
PROC     2 FICHERO NODO
         SET &TEMPFIL = &SYSUID..JMP.TMP
         IF &SYSDSN('&TEMPFIL') = OK THEN DELETE '&TEMPFIL'
         EDIT 'JMPDES.JCL.PROTOTIP(IBMXXVAX)' OLD NONUM CNTL
         TOP
         C * 999 'FFFFFF' '&FICHERO'  ALL
         TOP
         C * 999 'NNNNNN' '&NODO'     ALL
         SAVE '&TEMPFIL'
         END NOSAVE
         ALLOC FI(SYSUT1)   DA('&TEMPFIL')        SHR REUSE
         ALLOC FI(SYSUT2)   SYSOUT(A)  WRITER(INTRDR) REUSE
         ALLOC FI(SYSIN)    DUMMY                     REUSE
         ALLOC FI(SYSPRINT) DA(*)                     REUSE
         CALL 'SYS1.LINKLIB(IEBGENER)'
         FREE FILE(SYSUT1 SYSUT2 SYSPRINT SYSIN)
         DELETE '&TEMPFIL'
         WRITE ****************************************************
         WRITE ====> FICHERO : &FICHERO ENVIADO A &NODO  < OK >
         WRITE ****************************************************
ACABA: EXIT
END
```

Figura 4

EXCLUDE MEMBER=(miembro4,miembro5)

Cuando en la librería de salida existen los miembros especificados con la ficha *SELECT* no se copian los ficheros. No obstante, puede especificarse

Un fichero se encuentra catalogado cuando su nombre se encuentra en un fichero de la instalación

que reemplace al existente mediante el subparámetro R, tal y como muestra el siguiente ejemplo:

SELECT MEMBER=(MIEMBRO1,(miembro2,,R))

Cuando se desea que un miembro de la librería de entrada se copie con otro nombre sobre la librería de salida, se usará el segundo subparámetro:

SELECT MEMBER= ((miembro2,nom_nuevo,R),miembro3)

IEBUPDTE

Esta utilidad se usa para la creación y actualización de ficheros secuenciales o particionados a partir de registros lógicos de 80 caracteres.

Las operaciones posibles son:

- Incorporación de un nuevo miembro en un PDS.
- Sustitución de un miembro ya existente.
- Adición de un nuevo registro a un miembro o secuencial.
- Borrado de un registro lógico.
- Conversión de un fichero secuencial en un particionado.

Esta utilidad está algo desfasada, razón por la que no se van a describir sus comandos, y sólo se han descrito sus funciones por, si algún lector tuviera necesidad de hacer uso de ella, saber qué utilidad debe emplear.

JCL usado como prototipo en la clist de la figura U1.
Este JCL tiene por DSN=JMPDES.JCL.PROTOTIP(IBMXXVAX)

```
//JMPDESIE JOB  (12,345),'ENTORNO DESARROLLO',CLASS=T,MSGCLASS=X,
//             NOTIFY=JMPDES,MSGLEVEL=(0,0)
//*************************************************************
//SUBMITIR EXEC PGM=IEBGENER,COND=(0,LT)
//SYSUT2   DD SYSOUT=(A,INTRDR)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DATA,DLM=WB
//JMPPROIE JOB (34,567),'ENTORNO PRODUCCION',CLASS=K,MSGCLASS=X,
//         NOTIFY=JMPDES,MSGLEVEL=(0,0),RESTART=*
//** ************************************************************ **
//** ********** SE TRANSMITE EL FICHERO A DIGITAL ********* ITLIMITE  **
//** ************************************************************* **
//EJECUTAR EXEC PGM=IKJEFT01
//SYSEXEC  DD DSN=...,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  ... FICHAS DE CONTROL DE LA UTILIDAD A EJECUTAR
  SEND FFFFFF NNNNNN
/*
WB
```

Figura 5

PRÓXIMA ENTREGA

Puesto que el espacio no perdona, se aplaza para el próximo mes el resto de programas con un gran peso especifico en cualquier gran instalación, los programas *IDCAMS* e *IKJEFT01* o programa TSO para ejecutar en batch.

UTILIDAD

La utilidad que acompaña este mes al artículo es algo especial, ya que hace uso de la utilidad *IEBGENER* para submitir un JOB.

La utilidad de esta utilidad, valga la redundancia, radica en el hecho de saltarse los controles que establecen las instalaciones cuando se hace uso del comando *SUBMIT* de TSO, ya que escribe directamente en la cola interna de entrada de trabajos (*INTRDR: Internal Reader*).

La figura U1 muestra el listado de una CLIST o procedimiento de comandos TSO, que pide 2 datos: fichero y nodo. A continuación:

- Edita internamente el fichero o prototipo de JCL mostrado en la figura U2 y realiza las sustituciones de FFFFFF por el valor dado a la variable fichero, y NNNNNN por el valor dado a la variable Nodo.
- Salva el resultado de la edición sobre un fichero auxiliar.
- Aloca el fichero temporal dándole como *Ddname* SYSUT1.
- Aloca como salida la cola "A" de entrada de trabajos, asignándole la *Ddname* SYSUT2.
- Invoca al programa *IEBGENER*, que copia SYSUT1 sobre SYSUT2.
- Borra el fichero temporal.
- Devuelve un mensaje de fichero enviado.

Esta CLIST, suponiendo que se encontrara contenida en el fichero secuencial 'JMPDES.ENVIAR', ejecutando desde la línea de comandos: TSO EX'JMPDES.ENVIAR'.

VISUAL J++: JAVA CON SABOR A MICROSOFT

Carlos Gerardo Pérez Pérez

En este artículo se describe en la medida de los posible las prestaciones que ofrecerá en su versión final esta nueva herramienta de desarrollo, ya que por ser una versión Beta algunas de las funcionalidades que Microsoft piensa darle a esta herramienta aún no están disponibles

El pasado 12 de marzo Microsoft Corp. anunció la firma de un acuerdo con la compañía Sun Microsystems para licenciar la tecnología Java desarrollada por Sun con el objetivo de incluirla en productos Microsoft. La primera acción que Microsoft llevó a cabo fue incluir esta tecnología en la versión 3.0 de su nuevo navegador, el Internet Explorer, que estaba en desarrollo por aquel entonces.

Sin embargo, los esfuerzos de Mircrosoft en materia de Java han ido más lejos y el pasado verano anunciaron la distribución Beta de una herramienta de desarrollo de lenguaje Java a la que han bautizado con el nombre de MS Visual J++.

¿QUÉ ES EL MS VISUAL J++?

Visual J++ es un entorno visual de desarrollo de lenguaje Java, que se ha hecho tan popular por su características orientadas a objetos y de seguridad en la red Internet [1], que es capaz tanto de crear aplicaciones Java como applets que pueden ser integrados en documentos del WWW. Como prestación adicional tiene la posibilidad de que se pueden compilar en él controles que cumplan con la especificación ActiveX de Microsoft. Sin embargo, este tipo de desarrollos se verán limitados, ya que sólo serán visibles en aquellas plataformas que implementen esta tecnología (actualmente sólo Windows 95, Windows NT y Macintosh). El código de lenguaje Java que generan los asistentes de Visual J++ es completamente estándar con la especificación de Sun y el mismo código puede ser compilado con el JDK de Sun Microsystems sin ningún problema.

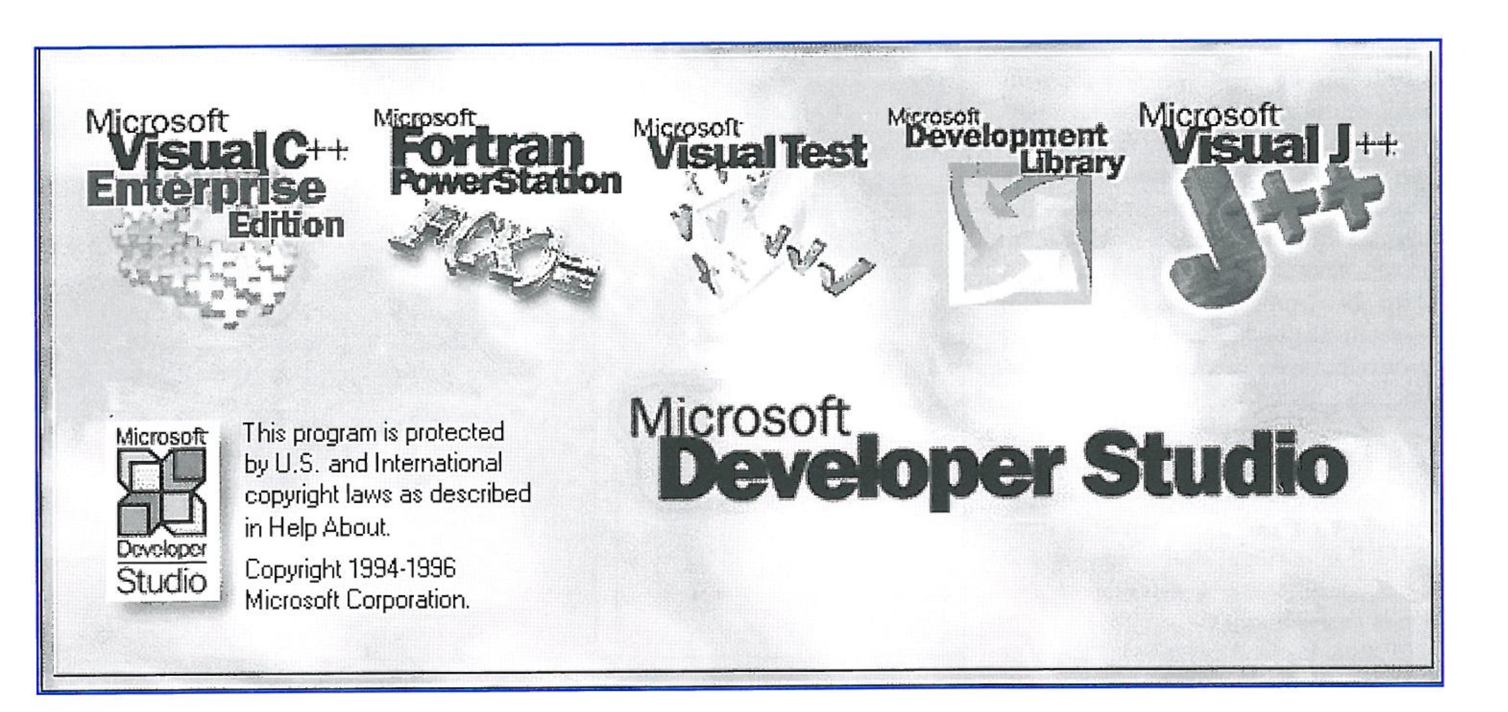
CARACTERÍSTICAS DE VISUAL J++

En el Visual J++, Microsoft integra el MS Developer Studio, que es el entorno integrado de desarrollo (IDE) con el que cuentan otras herramientas de Microsoft y que está formado por cuatro elementos principales:

- el asistente de desarrollo,
- el depurador gráfico,
- el compilador
- la documentación en línea.

Visual J++ es una herramienta bastante versátil y se pueden indicar como sus características más interesantes las siguientes:

- Minimizar el ciclo de edición - construcción - prueba de aplicaciones Java, ya que su compilador es capaz de ejecutar más de un millón de líneas de código por minuto para crear código Java para distintas plataformas.
- Solucionar problemas complejos rápidamente gracias el depurador visual de Java el cual permite depurar varios applets simultáneamente desde dentro del mismo navegador de Web, además de desensamblar código, colocar *breakpoints* en aplicaciones *multi-thread*, realizar un seguimiento del comportamiento de los distintos procesos, etc.
- Crear y probar applets y controles ActiveX simultáneamente.
- Acceder y cambiar el valor de variables o expresiones durante las pruebas, a través de su sistema DataTips
- Desplazarse visualmente usando el ClassView por el código como un conjunto de clases, métodos y variables relacionadas.



- Con la ayuda de los asistentes o AppWizards construir applets Java que sean multi-thread
- Construir aplicaciones de 32-bit bajo Windows 95 y Windows NT que puedan ser ejecutadas en otras plataformas.
- Explotar funcionalidades avanzadas instantáneamente con el extenso surtido de controles ActiveX existentes para un desarrollo rápido.

EL ENTORNO INTEGRADO DE DESARROLLO (IDE)

Como ya se ha mencionado Visual J++ utiliza como IDE el Developer Studio (ver Figura 1), el cual es el ambiente de desarrollo que también emplean otras herramientas de desarrollo de Microsoft como Visual C++, Microsoft Fortran PowerStation, Visual Test, etc.

Al lanzar Visual J++, en el Microsoft Developer Studio aparecen tres ventanas: La ventana del Project Workspace, el Editor, y la ventana de mensajes de la herramienta.

PROJECT WORKSPACE

En el Project Workspace hay tres ventanas principales que se pueden ver seleccionando la pestaña correspondiente. Dichas ventanas son: la ventana del ClassView, la del FileView y la del InfoView que permiten navegar rápidamente entre el código y la información que se requiera.

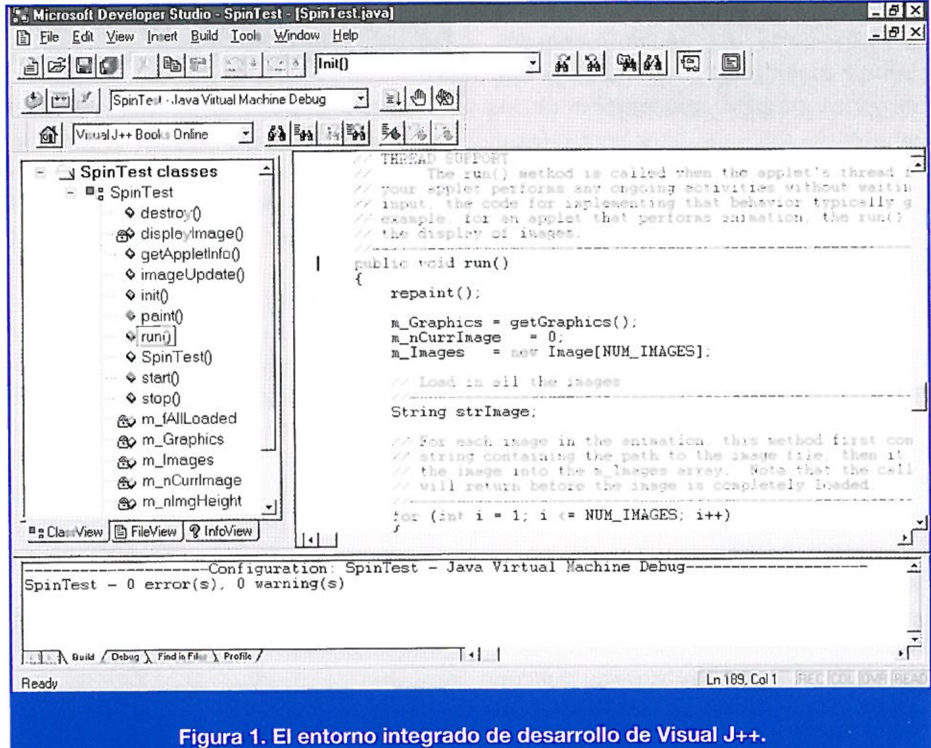


Figura 1. El entorno integrado de desarrollo de Visual J++.

- ClassView

En la ventana del ClassView se muestra el árbol del código del proyecto en el que se este trabajando. El código se muestra como un conjunto de clases, métodos y variables relacionados. Además, al realizar doble click sobre un método o variable es posible visualizar el código relacionado en el editor. En la ventana ClassView también se pueden agregar clases, métodos y variables y automáticamente verifica que se esté agregando o modificando todo conforme con la especificación del lenguaje Java.

- FileView

A través de la ventana se pueden ver los archivos que componen el proyecto en el que se esté trabajando, ya sea éste código Java, formato HTML, gráficos u otro formato relacionado con el proyecto.

- InfoView

En InfoView se puede ver la documentación en línea. Microsoft se ha comprometido a que entre ésta documentación en línea que incluirá Visual J++ estará: la especificación completa del lenguaje Java de Sun, la especificación del API, ejemplos y la inclusión del libro *Learn Java Now*. En la versión Beta aún no están disponible todos estos documentos pero en la versión final serán incluidos.

LOS ASISTENTES DE DISEÑO

Una de las mejores herramientas que se incluyen en Visual J++ son los asistentes de diseño o AppWizards con los que se pueden generar y ejecutar aplicaciones inmediatamente.

El Java Applet AppWizard (Figura 2) tiene una gran variedad de opciones para la generación de los applets, que incluyen la explicación sobre qué realiza cada parte del código generado, soporte multithread, soporte para animaciones simples, información de los parámetros que la página Web pasa al applet y la posibilidad de ejecutar el applet tanto como una aplicación al igual que como un applet, así mismo la posibilidad de establecer el tamaño del applet, etc.

Visual J++ también soporta otros AppWizards que faciliten el acceso a bases de datos DAO y RDO, así como la inclusión de AppWizards que sean producidos por otros fabricantes además de Microsoft.

EL DEPURADOR

El apartado más importante en una herramienta de desarrollo es la parte dedicada a trazar el código de los programas, y Visual J++ cuenta con un magnífico depurador que permite analizar cada punto del código y proporcionar de forma clara la información necesaria para su análisis y corrección.

Una de las mejores características de depurador es que se puede hacer el seguimiento de ciertas variables de interés dentro del programa, además de dar la posibilidad de alterar sus valores para ver qué ocurre con el comportamiento del programa (Figura 3). Otra de las innovaciones de Microsoft es lo que han llamado como DataTips, con la que con sólo colocar el cursor del ratón sobre una variable o parte de código remarcado en la ventana del código fuente, éste indicará el valor de la variable (Figura 4).

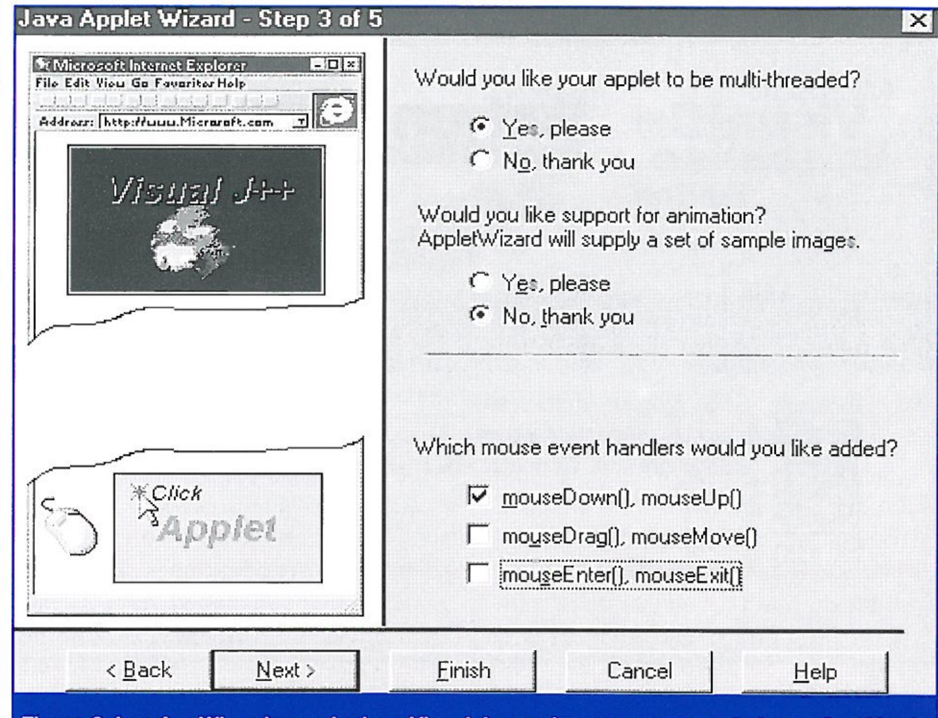


Figura 2. Los AppWizards que incluye Visual J++ mejoran enormemente la producción de applets para el WWW.

El depurador también cuenta con la utilidad que podríamos denominar depurador multithread, el cual a través de una caja de diálogo permite seleccionar uno de los procesos thread y analizarlo bien individualmente o junto a los demás. Y como los applets están hechos para trabajar dentro de un navegador, pues también se puede hacer la depuración con el applet ejecutándose en el navegador, pero eso sí, éste debe ser el MS Internet Explorer 3.0

LA EDICIÓN

- Editor de código

Microsoft, conocedor que la forma de edición de código es un elemento bastante personal del programador, permite que su editor sea altamente configurable. Es posible elegir el tipo de formato de edición, definición de macros, las tabulaciones, etc. Además, como reconoce la sintaxis

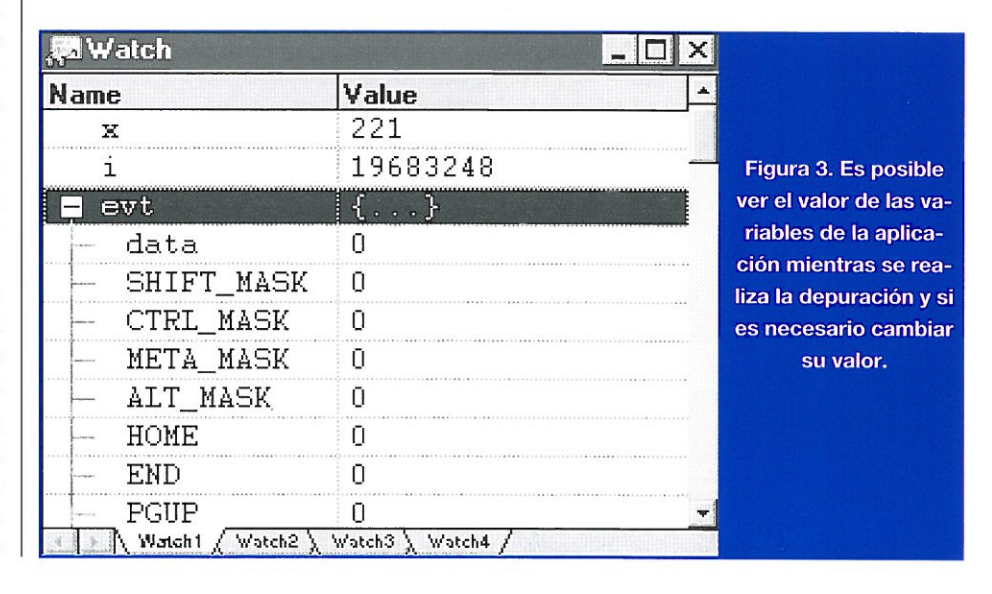


Figura 3. Es posible ver el valor de las variables de la aplicación mientras se realiza la depuración y si es necesario cambiar su valor.



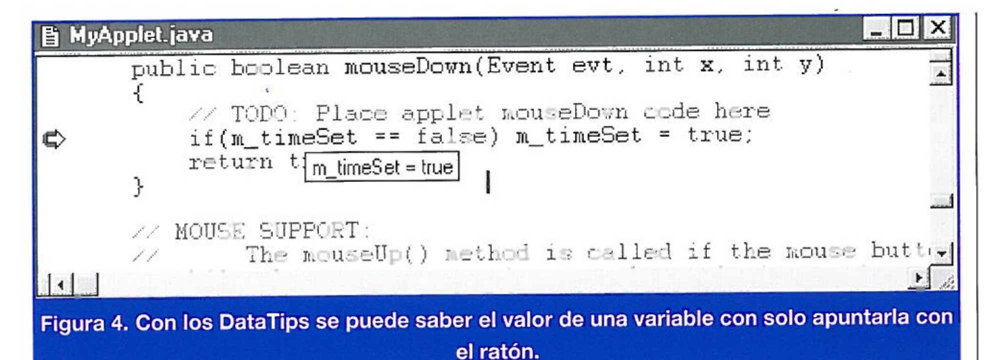


Figura 4. Con los DataTips se puede saber el valor de una variable con solo apuntarla con el ratón.

del código Java o del lenguaje HTML remarca en colores diferentes las palabras claves lo que facilita enormemente la lectura de código.

- Editor de Forms y Menús.
Una de las herramientas que más se agradecen a la hora de crear una aplicación son los editores de forms y menús, ya que permiten crear de una forma rápida la interfaz de la aplicación que en muchos de los casos es repetitiva y tediosa.

- Editor de Imágenes.
Como herramienta de desarrollo de aplicaciones para Internet no podía faltar un editor de imágenes pero el que incluye Visual J++ es muy limitado y poco práctico por lo que no se pueden hacer grandes cosas con él.

EL COMPILADOR Y EL SISTEMA DE CONSTRUCCIÓN

El compilador de Visual J++ y el sistema de construcción cumplen completamente con las especificaciones Java y las pruebas de conformidad de Sun para crear código para distintas plataformas. El compilador es capaz de procesar código a razón de un millón de líneas por minuto. Además, en cierta medida es un "caballo de troya" por parte de Microsoft, ya que también permite la integración de sus componentes ActiveX, que de una u otra manera quieren conseguir que la gente empiece a utilizar esta tecnología , y está claro que con Visual J++ es bastante natural la inclusión de los componentes ActiveX puesto que la llamada de los métodos a estos controles son como si estuvieran escritos en Java, veamos un ejemplo:

```
import com.ms.dao.Database
...
Database db = new Database (...);
db.OpenRecordset(...);
```

Los valores del constructor de proyectos pueden ser fácilmente modificados y personalizados (Figura 5). Se puede especificar si es una aplicación o un applet lo que se está construyendo, si se quiere depurar utilizando el Internet Explorer 3.0 y qué parámetros son los que le debe pasar la página HTML al applet, etc.

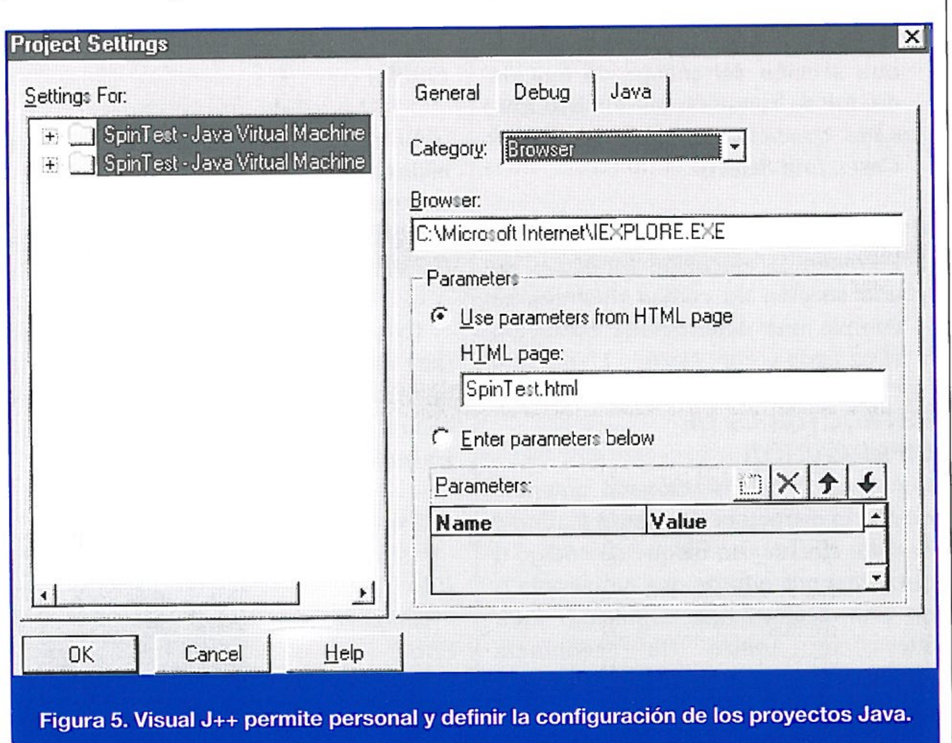


Figura 5. Visual J++ permite personal y definir la configuración de los proyectos Java.

REQUERIMIENTOS DE VISUAL J++

Para ejecutar Visual J++ al menos es necesario lo siguiente :

- Procesador 486 o superior.
- Sistema operativo: MS Windows 95 o MS Windows NT Workstation 4.0 o posterior.
- 8 MB de memoria,(12 MB recomendado) si se ejecuta en Windows 95; 16 MB (20 MB recomendado) si se ejecuta en Windows NT Workstation.
- Espacio de disco duro: Instalación típica 20 MB - Instalación mínima: 14 MB con herramientas ejecutándose desde el CDROM.
- Lector de CDROM.
- Al menos monitor VGA pero se recomienda SVGA.
- Ratón Microsoft o compatible.

Como se ha visto a lo largo del artículo MS Visual J++ es una herramienta muy completa, que facilita en gran medida con sus componentes la creación de applets y aplicaciones Java dentro de las plataformas de MS Windows. Y tiene todos los ingredientes para convertirse en el patrón de desarrollos Java. Por ahora se puede obtener la versión Beta y mayor información en las páginas Web de microsoft en http://www.microsoft.com/visualj

Referencias:

[1] Java: La Revolución Internet, Fernando J. Echevarrieta, Año 3, No 22, de Sólo Programadores



CONCEPTOS DE PROGRAMACIÓN

Juan Manuel y Luis Martín

En los artículos anteriores se han visto a grandes rasgos los distintos elementos que componen una aplicación desarrollada con Visual Basic. Sin embargo, para que ésta funcione en la forma deseada por el programador, es necesario conjuntarlos y controlar su ejecución. En este artículo se analizará la forma en que se ejecuta el código dentro de los procedimientos.

Inicialmente, las instrucciones contenidas en un procedimiento se ejecutan de forma secuencial, una detrás de otra, desde la primera hasta la última. Sin embargo, puede interesar que el flujo de una aplicación no discurra de ese modo, sino que ciertas partes del mismo no se ejecuten, mientras que otras lo hagan varias veces. Para alterar el flujo de un programa se utilizan un conjunto de sentencias especiales, denominadas sentencias de control. Este tipo de sentencias se utiliza para construir estructuras de control, que permitirán repetir parte del código hasta que suceda algo, realizar bifurcaciones en el mismo en función de un resultado, etc.

Las estructuras de control disponibles en Visual Basic pueden clasificarse en dos tipos, según su finalidad:

- Estructuras de bifurcación o toma de decisión: Permiten ejecutar una u otra sección del código, en función de que se cumpla o no una determinada condición (*If...End If* y *Select Case...End Select*).

- Estructuras repetitivas o de bucle: Permiten la ejecución repetitiva de una sección de código mientras se cumpla una determinada condición (*Do...Loop* y *For...Next*)

ESTRUCTURAS DE BIFURCACIÓN

Como ya se ha mencionado anteriormente, la estructura de control *If...End If* permite ejecutar un bloque de código u otro, dependiendo de que se cumpla o no una determinada condición. Para ello, se utiliza la sentencia *If...Then...Else*, que presenta varios formatos, algunos de ellos muy completos.

La sintaxis del formato más sencillo de esta sentencia puede representarse de la siguiente forma:

```
If Condición Then
  [Instrucciones]
End If
```

Cuando durante la ejecución se llega a la sentencia *If*, se evalúa la condición lógica que la acompaña. Si el resultado es *True*, se ejecutan todas las instrucciones que se encuentren a continuación, hasta la cláusula *End If*. En caso contrario, estas sentencias no se ejecutan, pasando a ejecutarse la instrucción siguiente a *End If*:

```
If Fecha != Date Then
  Print "La fecha no se corresponde con
la actual"
End If

If False Then
  'El programa nunca pasará por aquí
End If
```

Si dentro de la estructura sólo se incluye una única instrucción, la estructura puede ser escrita en una sola línea, omitiendo la cláusula *End If*.

```
If Condición Then Instrucción
```

Existe un segundo formato que permite ejecutar un grupo de instrucciones cuando la condición se cumpla, y otro distinto cuando no se cumpla. Su sintaxis es la siguiente:

```
If Condición Then
  [Instrucciones]
[Else
  [Instrucciones]]
End If
```

En este caso, cuando el resultado de evaluar *Condición* es el valor *False*, se

ejecutan todas las instrucciones que se encuentran entre *Else* y *End If*.

If Fecha = Date Then
Print "La fecha se corresponde con la actual"
Else
Print "La fecha NO se corresponde con la anterior"
End If

If True Then
'El programa SIEMPRE pasará por aquí
Else
'El programa NUNCA pasará por aquí
End If

Igual que ocurría con el formato más sencillo, cuando sólo hay una instrucción en cada una de las secciones de la estructura, ésta puede escribirse en una sola línea.

If Condición Then Instrucción Else Instrucción

Finalmente, existe un último formato que permite ejecutar un número indefinido de grupos de instrucciones, atendiendo al cumplimiento de diversas condiciones. Este es el formato mas completo y su sintaxis es la siguiente:

If Condición1 Then
[Instrucciones1]
[ElseIf Condición2 Then
[Instrucciones2]]
[ElseIf Condición3 Then
[Instrucciones3]]
...
[Else
[InstruccionesN]]
End If

La figura 1 muestra un diagrama de flujo con el funcionamiento de esta estructura. Así, cuando se llega a la sentencia *If*, se evalúa *Condición1*. Si el resultado es *True*, se ejecuta el grupo *Instrucciones1* y se pasa a la siguiente instrucción a *End If*. Si por el contrario, el resultado es *False*, se evalúa *Condición2*. Si el resultado es *True*, se ejecuta el grupo *Instrucciones2* y se pasa a la siguiente instrucción a *End If*. Si el resultado es *False*, se evalúa *Condición3*, y así sucesivamente. Por último, si el resultado de todas las evaluaciones es *False*, se ejecutan el grupo *InstruccionesN* de la cláusula *Else*, finalizando la ejecución de la estructura.

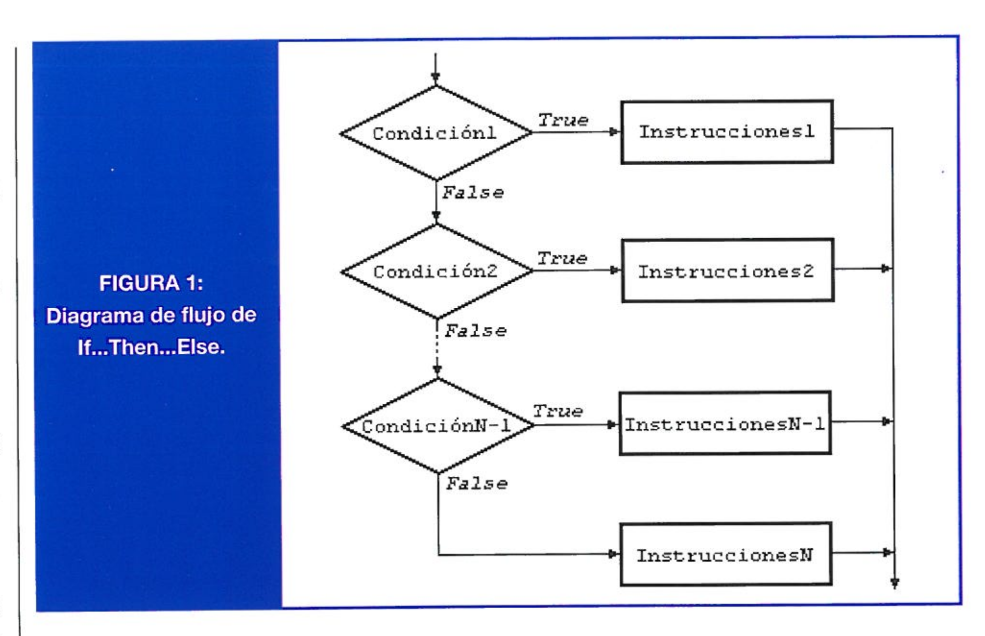


FIGURA 1: Diagrama de flujo de If...Then...Else.

If Numero = 1 Then
Print "El valor es 1"
ElseIf Numero = 2 Then
Print "El valor es 2"
Else
Print "El valor es mayor de 2"
End If

Visual Basic también dispone de una función que opera de forma muy similar a la estructura *If...End If*, aunque su misión es devolver valores en vez de ejecutar instrucciones. Se trata de la función *IIf*, que devuelve un valor cuando se cumple una condición lógica y otro distinto cuando no se cumple. Su sintaxis es la siguiente:

IIf(Condición, ExpVerdadera, ExpFalsa)

Al ejecutarse la función, primero se evalúa *Condición*. Si el resultado es *True*, la función devuelve el valor de *ExpVerdadera*. Si, por el contrario, el resultado es *False*, la función devuelve el valor de *ExpFalsa*. El resultado devuelto por la función puede ser de cualquier tipo de datos, ya que *ExpVerdadera* y *ExpFalsa* también pueden serlo.

Print "La respuesta es " & IIf(Resp, "Si", "No")

LA ESTRUCTURA CASE

La estructura de control *Select Case...End Select* permite ejecutar un grupo de instrucciones de entre varios posibles, dependiendo del valor tomado por una determinada expresión de evaluación. Su sintaxis y su modo de operar son muy similares al formato más completo de *If...End If*. La sintaxis puede resumirse de la siguiente forma:

La sentencia If...Then...Else presenta varios formatos, algunos de ellos muy completos

Select Case Expresión
[Case ListaComprobación1
[Instrucciones1]]
[Case ListaComprobación2
[Instrucciones2]]
...
[Case Else
[InstruccionesN]]
End Select

Cuando se llega a la sentencia *Select Case* se evalúa *Expresión*, cuyo resultado puede ser un valor de cualquier tipo de datos. Una vez obtenido dicho valor, se pasa el control a la primera cláusula

Case. Si el valor se corresponde con alguno de los indicados en *ListaComprobación1* se ejecuta el grupo *Instrucciones1*, pasando a ejecutarse la siguiente instrucción a *End Select*. Si el valor no se encuentra en *ListaComprobación1*, se pasa el control al segundo *Case*, y se comprueba si el valor se encuentra en la *ListaComprobación2*. En caso afirmativo, se ejecuta el grupo *Instrucciones2*, pasando a ejecutarse la siguiente instrucción a *End Select*. Si el valor no se encuentra en *ListaComprobación2*, se pasa el control al siguiente *Case*, y así sucesivamente. Por último, si el valor devuelto por *Expresión* no se encuentra en ninguna de las listas de comprobación se ejecuta el grupo *InstruccionesN*, comprendidas entre *Case Else* y *End Select*, finalizando la ejecución de la estructura.

Las listas de comprobación de las cláusulas *Case* pueden estar constituidas por un número indefinido de expresiones, separadas unas de otras por comas. Evidentemente, para que éstas puedan equipararse el resultado de la expresión de evaluación sin que se produzca ningún error, ambas deben pertenecer al mismo tipo de datos.

```
Select Case WeekDay(Date)
  Case 2
    Print "Hoy es Lunes"
  Case 3
    Print "Hoy es Martes"
  Case 4
    Print "Hoy es Miércoles"
  Case 5
    Print "Hoy es Jueves"
  Case 6
    Print "Hoy es Viernes"
  Case 1, 7
    Print "Hoy es fin de semana"
End Select
```

Las expresiones de las listas de comprobación, además de expresiones normales de cualquier tipo de datos, pueden presentar otros formatos. Es posible indicar un rango de valores haciendo uso de la cláusula *To*, de forma similar a como se realiza en la declaración de matrices.

ExpresiónMin To ExpresiónMax

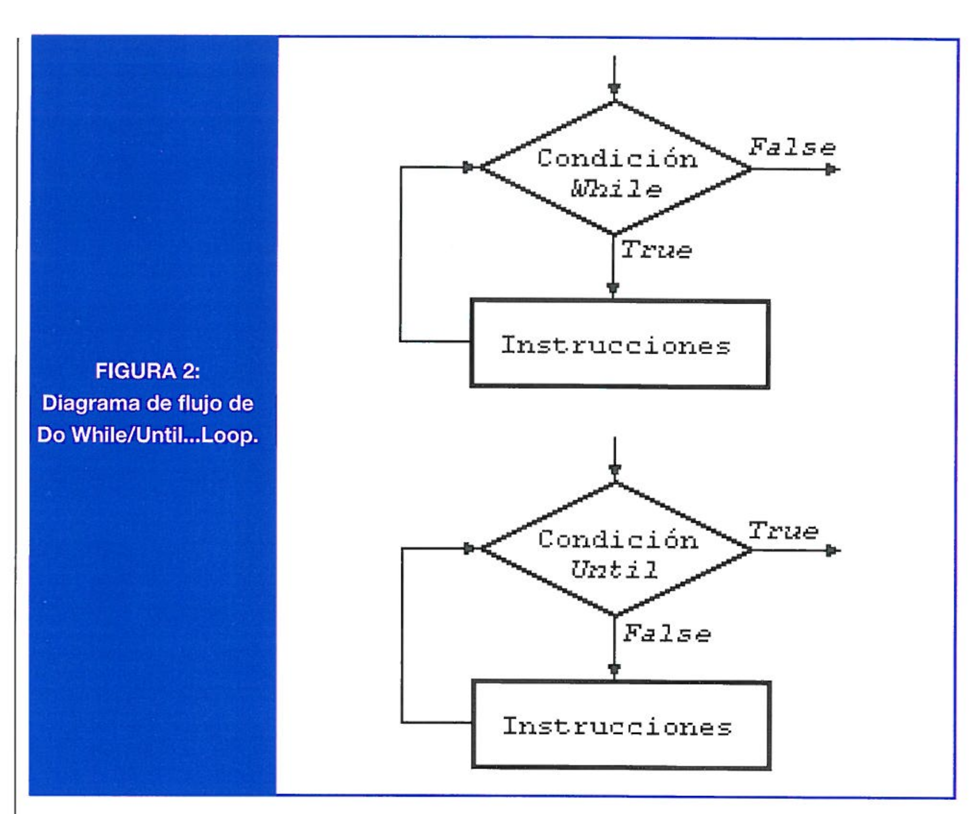


FIGURA 2: Diagrama de flujo de Do While/Until...Loop.

En este caso, la comprobación resultará positiva cuando el resultado de la expresión de evaluación se encuentre entre los valores determinados por *ExpresiónMin* y *ExpresiónMax*, ambos inclusive.

También es posible indicar una condición de comparación con un determinado valor mediante el uso de una cláusula especial. Se trata de la cláusula *Is*, que debe ir siempre acompañada de uno de los operadores relacionales (=, <, >, <=, >=, <>).

Is OpRelacional ExpComparación

En este caso, la comprobación resultará positiva cuando el resultado de comparar la expresión de evaluación con *ExpComparación* sea verdadero, es decir, cuando devuelva *True*. El cuadro 1 muestra dos fragmentos de código que ilustran el funcionamiento de las cláusulas *To* e *Is*.

Al igual que ocurría con la función *IIf*, Visual Basic también dispone de una función que opera de forma muy similar a la estructura *Select Case...End Select*. Se trata de la función *Choose*, que devuelve distintos valores en función del valor de una expresión. Su sintaxis es la siguiente:

Choose(ExpÍndice, Expresión1 [, Expresión2 [,... [, ExpresiónN]]])

Al ejecutarse la función se evalúa la expresión *ExpÍndice*, cuyo valor debe ser un número comprendido entre 1 y el número de opciones especificadas en la función. Si el valor es 1, la función devuelve el resultado de evaluar *Expresión1*. Si es 2, se devuelve el resultado de evaluar *Expresión2*, y así sucesivamente. Los valores de estas expresiones son tomados como de tipo *Variant*, por lo que el valor devuelto por la función también lo será.

La función IIf opera de forma muy similar a la estructura If...End If

Print Choose(Sgn(Num)+2, "Negativo", "Cero", "Positivo")

LOS BUCLES DO...LOOP

La estructura *Do...Loop* se utiliza para ejecutar un grupo de instrucciones un número indefinido de veces en función

del cumplimiento de una determinada condición lógica. Dicha condición será la que permita repetir de nuevo la ejecución del grupo de instrucciones o, en su caso, abandonar la estructura.

La construcción de este tipo de estructuras de bucle en Visual Basic se realiza mediante la sentencia *Do...Loop*. Esta sentencia presenta dos formatos distintos en función del lugar elegido para realizar la evaluación de la condición. Si se desea realizar la evaluación al principio del bucle, la sintaxis será la siguiente:

```
Do [{While|Until} Condición]
  [Instrucciones]
  [Exit Do]
  [Instrucciones]
Loop
```

Como puede apreciarse, además de disponer de dos formatos, es posible utilizar dos cláusulas distintas para realizar la evaluación de la condición: *While* (mientras) o *Until* (hasta). La utilización de cada una de estas dos palabras clave determina el momento de salida del bucle. La figura 2 muestra un diagrama de flujo con ambos casos.

Cuando durante la ejecución de la aplicación se llega a una sentencia *Do*, acompañada de la cláusula *While*, se evalúa *Condición*. Si el resultado es *True*, se ejecutarán todas las instrucciones entre *Do* y *Loop*. Al llegar a la cláusula *Loop*, vuelve a ejecutarse la sentencia *Do*, evaluándose de nuevo la condición lógica. Si el resultado es otra vez *True*, se vuelve a ejecutar el grupo de instrucciones. Esta operación se repite hasta que la condición devuelva un valor *False*. En ese momento, se abandona el bucle, pasando a ejecutarse la siguiente instrucción a *Loop*.

```
Num = 0
Do While Num < 5
  Num = Num + 1
Loop
Print Num          'Resultado: 5
```

Si, por el contrario, durante la ejecución de la aplicación se llega a una sentencia *Do*, acompañada de la cláusula *Until*, el funcionamiento es inverso. En primer lugar se evalúa *Condición*. Si el resultado es *False*, se ejecutarán todas las instrucciones entre *Do* y *Loop*.

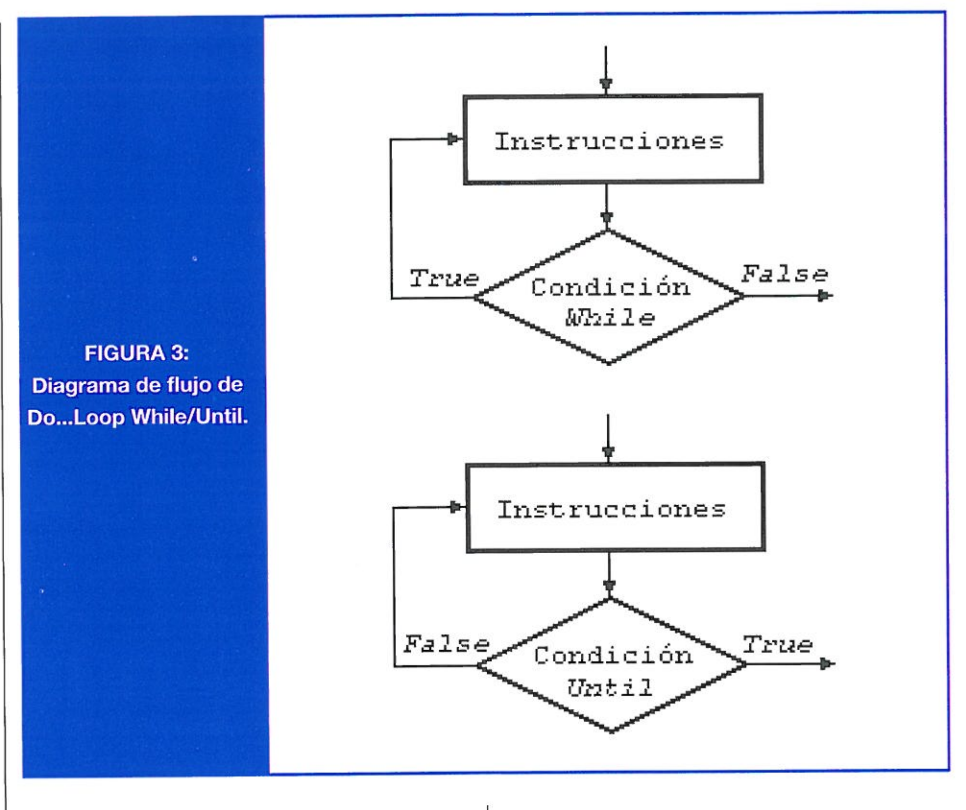


FIGURA 3: Diagrama de flujo de Do...Loop While/Until.

Seguidamente, vuelve a ejecutarse la sentencia *Do*, evaluándose de nuevo la condición. Si el resultado es otra vez *False*, se vuelve a ejecutar el grupo de instrucciones. Esta operación se repite hasta que la condición devuelva *True*. En ese momento se abandona el bucle, pasando a ejecutarse la siguiente instrucción a *Loop*.

```
Num = 0
Do Until Num = 3
  Num = Num + 1
Loop
Print Num          'Resultado: 3
```

Por el contrario, cuando se desea realizar la evaluación al final del bucle, el funcionamiento es similar. La única diferencia consiste en que el grupo de instrucciones se ejecutará siempre al menos una vez, ya que la evaluación de la condición se realiza al final. La sintaxis es la siguiente:

```
Do
  [Instrucciones]
  [Exit Do]
  [Instrucciones]
Loop [{While|Until} Condición]
```

La figura 3 muestra los dos casos posibles para este formato. Así, cuando el programa llega a una sentencia *Do*, sin cláusulas que la acompañen, se ejecuta el grupo de instrucciones entre *Do* y *Loop*. A continuación se evalúa la condición lógica de la cláusula *Loop*. Si ésta viene acompañada de una cláusula *While*, y el resultado de la evaluación es *True*, se vuelve a ejecutar el grupo de instrucciones. Lo mismo ocurre si viene acompañada de *Until* y el resultado de la evaluación es *False*. Si, por el contrario, la cláusula es *While* y el resultado *False*, o la cláusula es *Until* y el resultado *True*, se abandona el bucle, pasando a ejecutarse la siguiente instrucción a *Loop*.

Las listas de comprobación de las cláusulas Case pueden estar constituidas por un número indefinido de expresiones

```
Cad = ""
Do
  Cad = Cad & "*"
Loop While Len(Cad) < 5
Print Cad          'Resultado: *****

Cad = ""
Do
  Cad = Cad & "*"
Loop Until Len(Cad) = 3
Print Cad          'Resultado: ***
```

Dentro del grupo de instrucciones que se encuentran incluidas en la estructura debe variarse de alguna forma el resultado de la evaluación de la condición. Si esto no se realiza, el programa será incapaz de abandonar la estructura, ejecutándose indefinidamente el bucle y haciendo que el ordenador quede "colgado".

```
Num = 1
Do While Num < 5
  'El ordenador queda colgado
Loop
```

Entre las instrucciones que componen el cuerpo de la estructura *Do...Loop* es posible incluir una instrucción que permite abandonar la ejecución del bucle de forma anticipada, independientemente de si la condición de salida se cumple o no. Se trata de la sentencia *Exit Do*, cuya ejecución fuerza la salida automática de la estructura, saltándose las instrucciones que falten por ejecutarse dentro del bucle y pasando a la siguiente instrucción a *Loop*.

El siguiente fragmento de código permite buscar un determinado valor dentro de una matriz unidimensional:

```
Ind = LBound(Matriz)
Do While Ind <= UBound(Matriz)
  If Matriz(Ind) = Valor
    Exit Do
  End If
  Ind = Ind + 1
Loop
If Ind > UBound(Matriz) Then
  Print "El valor NO se encuentra"
Else
  Print "El valor se encuentra en la posición " & Str(Ind)
End If
```

CUADRO 1

Ejemplo de utilización de las cláusulas *To* e *Is*.

```
'Determinación del tipo de un carácter
Select Case Letra$
  Case "A", "E", "I", "O", "U", "a", "e", "i", "o", "u"
    Print "Es una vocal"
  Case "A" To "Z", "a" To "z"
    Print "Es una consonante"
  Case "0" To "9"
    Print "Es una cifra"
  Case Else
    Print "Es un carácter especial"
End Select

'Determinación del signo de un número
Select Case Numero
  Case Is < 0
    Print "El valor es negativo"
  Case Is > 0
    Print "El valor es positivo"
  Case Else
    Print "El valor es 0"
End Select
```

Mediante la utilización de la sentencia *Exit Do* es posible abandonar estructuras de bucle indefinidas. Este tipo de estructuras se construyen haciendo que la condición lógica para repetir el bucle se cumpla siempre.

```
Do While True
...                'Debe incluir Exit Do para poder salir
Loop
```

Dentro del grupo de instrucciones que componen el cuerpo de una estructura *Do...Loop* es posible incluir otra estructura de este tipo. A su vez, dentro de ésta última es posible incluir otra, y así sucesivamente. A este tipo de construcción se le denomina anidamiento. De esta forma, para cada una de las iteraciones del bucle externo, el bucle interno se ejecutará completamente.

```
Ext = 1
Do While Ext < = 5
  Int = 1
  Do While Int <= Ext
    Print "*";
    Int = Int + 1
  Loop
  Print
  Ext = Ext + 1
Loop
```

CUADRO 2

Obtención del Factorial de un número.

```
'Forma ascendente
Factorial = 1
For I = 1 To Numero
  Factorial := Factorial * I
Next I
Print "El factorial de " & Str(Numero) & " es " & Str(Factorial)

'Forma descendente
Factorial = 1
For I = Numero To 1 Step -1
  Factorial := Factorial * I
Next I
Print "El factorial de " & Str(Numero) & " es " & Str(Factorial)
```

En el ejemplo anterior, la ejecución del bucle exterior está controlada por la variable *Ext*. Para cada valor de ésta se ejecutará completo el bucle interior que, a su vez, está controlado por la variable *Int*. En cada iteración del bucle exterior, el bucle interior se ejecutará tantas veces como marque la variable *Ext*. El resultado de este ejemplo será el siguiente:

```
*
**
***
****
*****
```

Uno de los formatos de la estructura *Do...Loop* más frecuentemente utilizado es aquel en que la evaluación de la condición se realiza al principio del bucle mediante la cláusula *While*. Por ello, Visual Basic dispone de otra sentencia que permite realizar la misma estructura. Se trata de la sentencia *While...Wend*, cuya sintaxis es la siguiente:

```
While Condición
  [Instrucciones]
Wend
```

CUADRO 3

Ejemplo de bucles *For...Next* anidados.

```
'Búsqueda de un valor en una matriz multidimen-
sional
Encontrado = False
For I = LBound(Matriz, 1) To UBound(Matriz, 1)
  For J = LBound(Matriz, 2) To UBound(Matriz, 2)
    For K = LBound(Matriz, 3) To UBound(Matriz, 3)
      If Matriz(I, J, K) = Valor Then
        Encontrado = True
      End If
    Next K
  Next J
Next I
If Encontrado Then
  Print "El valor ha sido encontrado"
Else
  Print "El valor NO ha sido encontrado"
End If
```

El funcionamiento de esta estructura es idéntico al de la estructura *Do...Loop* anteriormente mencionada. Sin embargo, para este caso no es posible utilizar una sentencia de salida automática del bucle, como *Exit Do*. El siguiente ejemplo permite listar todos los elementos de una matriz.

```
Ind = LBound(Matriz)
While Ind <= UBound(Matriz)
  Print Matriz(Ind)
  Ind = Ind + 1
Wend
```

LOS BUCLES FOR...NEXT

La estructura *For...Next* también se utiliza para ejecutar un grupo de instrucciones repetidamente. A diferencia de la estructura anterior, ésta requiere establecer de antemano el número de veces que se ejecutará el bucle. Para ello se utiliza una variable de control, cuyo valor se irá incrementando o decrementando en cada iteración.

La construcción de este tipo de estructuras de bucle en Visual Basic se realiza mediante la sentencia *For...Next*. Su sintaxis es la siguiente:

```
For Variable = ExpInicio To ExpFin
[Step ExpIncremento]
  [Instrucciones]
  [Exit For]
  [Instrucciones]
Next [Variable]
```

Cuando el programa llega a una sentencia *For*, la variable de control especificada en Variable toma el valor indicado en *ExpInicio*. A continuación se ejecutan las instrucciones que se encuentran entre *For* y *Next*. Una vez ejecutadas, la variable de control se incrementará en el valor indicado en *ExpIncremento*, volviendo a ejecutarse el mismo grupo de instrucciones. La operación se repite hasta que la variable de control supera el valor indicado en la *ExpFin*, terminando la ejecución de la estructura y pasando a ejecutarse la siguiente instrucción a *Next*. Cuando se omite la cláusula *Step* el valor por defecto de *ExpIncremento* será 1.

Si se indica en *ExpInicio* un valor menor que en *ExpFin*, la variable de control recorrerá el rango de valores en sentido inverso. En este caso será necesario indicar un valor negativo en *ExpIncremento*, ya que de lo contrario el bucle nunca se ejecutará. El cuadro 2 muestra dos fragmentos de código que permiten calcular el factorial de un número de forma ascendente y descendente.

Al igual que las estructuras del tipo *Do...Loop*, también es posible utilizar una sentencia que permita abandonar el bucle sin que éste se complete. En este caso, la sentencia es *Exit For*. Cuando se ejecuta esta sentencia se produce la salida automática de la estructura, saltándose así todas las instrucciones que falten por ejecutarse dentro del bucle y pasando a la siguiente instrucción a *Next*.

También es posible anidar estructuras *For...Next*, de forma idéntica al caso de la estructura *Do...Loop*. Las estructuras *For...Next* anidadas son especialmente útiles para el tratamiento de matrices multidimensionales. El cuadro 3 muestra un fragmento de código que recorre una matriz tridimensional para comprobar si contiene un determinado valor.

LOS SALTOS INCONDICIONALES

Aunque su uso no es en absoluto recomendable, Visual Basic dispone también de una sentencia de salto incondicional. Se trata de la sentencia *GoTo*, que permite hacer un salto en el flujo del programa hasta una línea de código especificada. Su sintaxis es *GoTo Línea*.

En el argumento *Línea* debe especificarse el número de línea o la etiqueta de la línea de código a la que se desea saltar. La sentencia *GoTo* sólo permite saltar a líneas de código que se encuentren dentro del mismo procedimiento:

```
GoTo 100
...
100

GoTo Aquí
...
Aquí:
```

Existe también otra sentencia similar que permite realizar saltos incondicionales a diversas líneas de código en función del resultado de la evaluación de una expresión. Su sintaxis es la siguiente:

On Expresión GoTo ListaLíneas

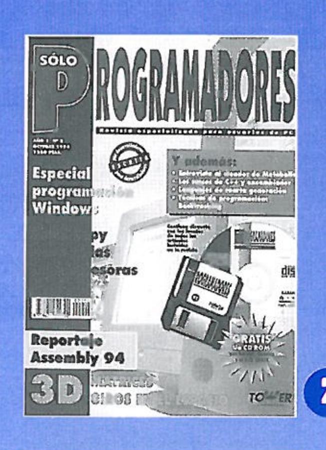
Cuando el programa llega a una sentencia de este tipo, se evalúa *Expresión*. El resultado debe ser un número comprendido entre 0 y el número de opciones especificadas en *ListaLíneas*. Si es 0, la ejecución continúa en la instrucción siguiente a *On*. Si valor es 1, se produce un salto incondicional a la línea especificada en el primer lugar de *ListaLíneas*. Si es 2, el salto se realizará a la línea especificada en la segunda posición, y así sucesivamente. Evidentemente, *ListaLíneas* estará constituida por una lista de números de línea o etiquetas, separadas entre sí por comas, hasta un máximo de 255.

```
On Numero GoTo Aquí, Allí
...
Aquí:
...
Allí:
...
```

¡Que no te falte ni uno!



1



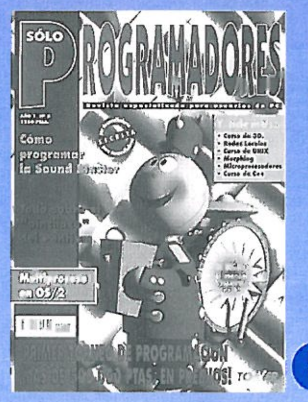
2



3




4



5



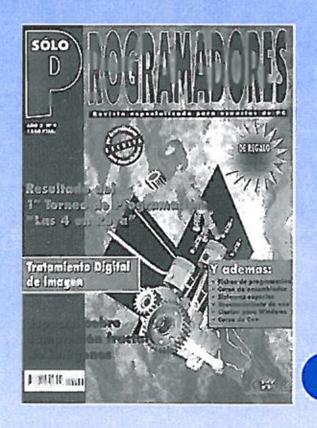
6



7




8



9



10



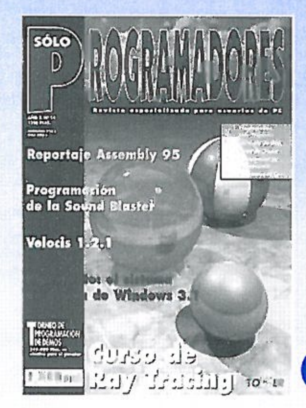
11




12



13



14



15

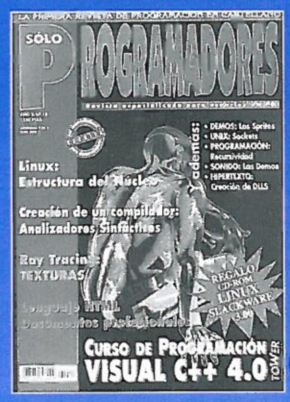


16

•NÚMEROS ATRASADOS•NÚMEROS ATRASADOS•



17



18



19



20



21



22



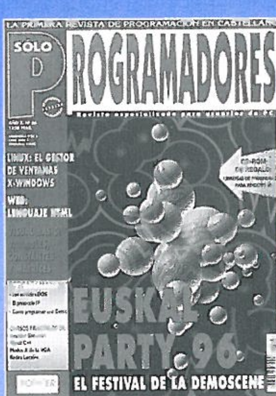
23



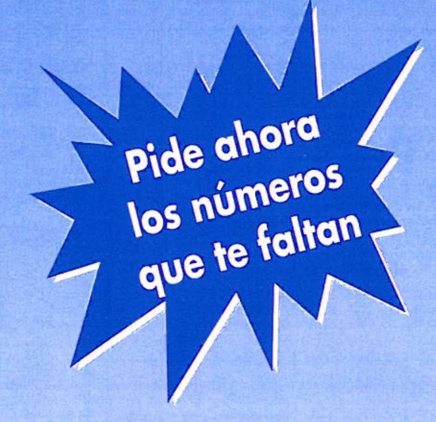
24



25



26



COMPLETA YA TU COLECCIÓN

Solicite los números que le falten enviando este cupón por correo o fax (91) 661 43 86, o llamando al teléfono (91) 661 42 11*. Horario lunes a jueves de 9 a 14 y 15 a 18h. y viernes de 9 a 15h.

SOLICITO LOS SIGUIENTES NÚMEROS ATRASADOS DE PC MEDIA

☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9 ☐10 ☐11 ☐12 ☐13 ☐14 ☐15 ☐16 ☐17 ☐18 ☐19 ☐20 ☐21 ☐22 ☐23 ☐24 ☐25 ☐26 Importe.......................................(1.250 pts. cada uno)

Nombre y apellidos..Domicilio...

Población..C.P ....Provincia.........................Telf...................................Profesión......................

FORMA DE PAGO:

☐ Con cargo a mi tarjeta VISA nº ____ ____ ____ ____

Fecha de caducidad de la tarjeta.................Nombre del titular, si es distinto...

☐ Contra-reembolso del importe más gastos de envío.

☐ Cheque a nombre de TOWER COMMUNICATIONS S.R.L., que adjunto.

☐ Giro Postal (adjunto fotocopia del resguardo).

Firma:

Rellena este cupón y envíalo a:
TOWER COMMUNICATIONS SRL
C/ Aragoneses, 7
28100 Pol. Ind. ALCOBENDAS (Madrid)

CONTENIDO DEL CD-ROM

SQL ANYWHERE Y VISUAL COMPONETS

El presente número de Sólo Programadores viene acompañado de un CD-ROM que contiene la versión de evaluación para un periodo de 60 días de Sybase SQL Anywhere y la versión de prueba de Visual Components.

Sybase SQL Anywhere es un sistema cliente/servidor de acceso a datos mediante SQL. El CD-ROM contiene el software servidor, el motor autónomo, el software cliente y el motor autónomo con SQL remoto.

Los Visual Components son un conjunto de controles ActiveX pensados para potenciar las aplicaciones de Windows 95. Entre otros, se incluye un componente de hoja de cálculo compatible con Excel y un componente de gráficos de negocios.

SYBASE SQL ANYWHERE

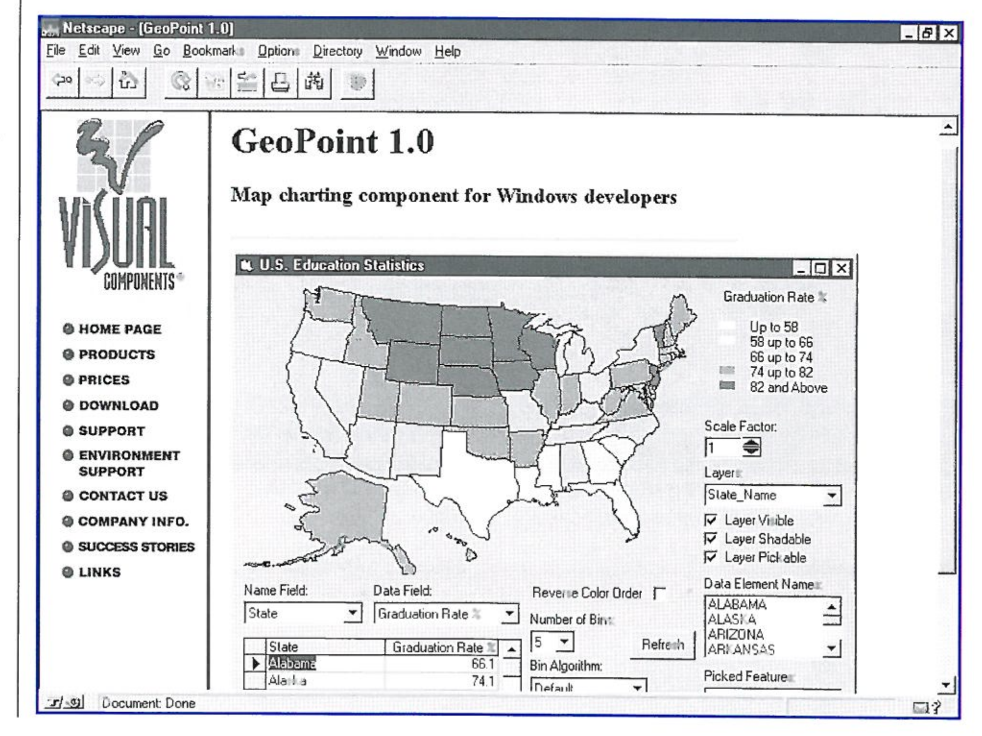
Los archivos correspondientes a SQL Anywhere están dentro del directorio \SQLANY del CD-ROM. Puede ser instalado tanto en Windows como en DOS, OS/2 y NetWare (este último sólo como servidor).

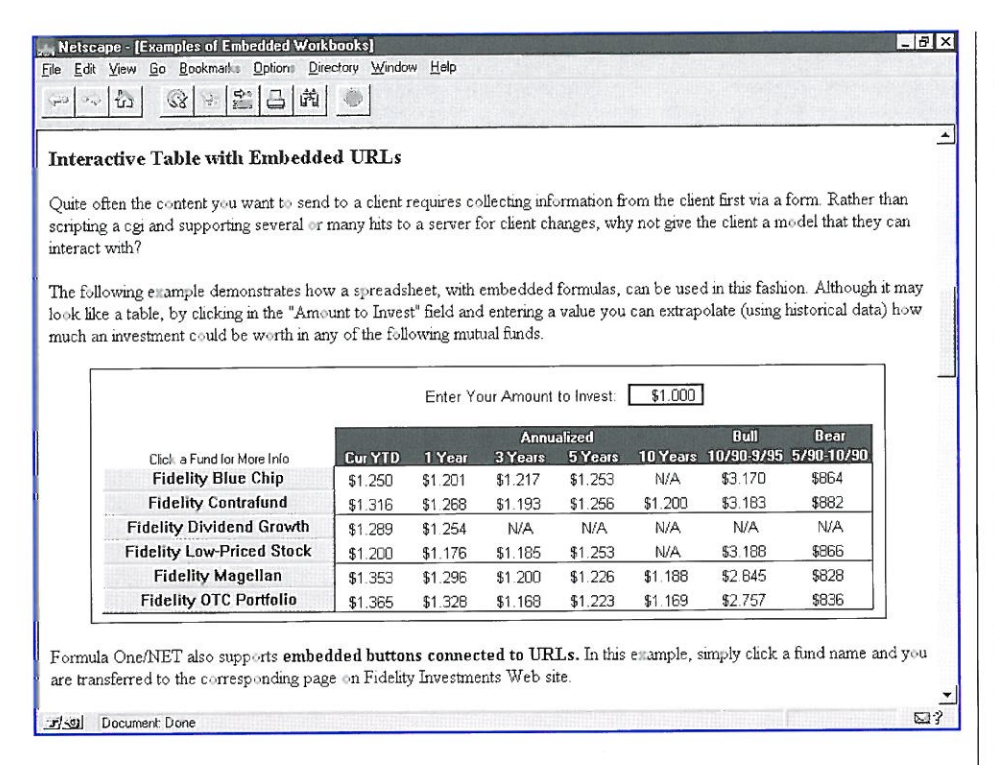
Para instalar las versiones de Windows o bien la de DOS se debe ejecutar el programa SETUP.EXE. Por el contrario, si lo que se desea es instalar la versión de OS/2 se debe ejecutar INSTALL.EXE. La versión de NetWare debe ser instalada desde una máquina cliente. El ejecutable de instalación (SETUP.EXE o INSTALL.EXE) depende del sistema operativo utilizado por dicha máquina.

El programa de instalación preguntará cuál de los cuatro componentes se desea instalar:

- Servidor (incluye el motor autónomo con SQL remoto).
- Cliente.
- Motor autónomo.
- Motor autónomo con SQL remoto.

En la versión de Windows, si se decide instalar el servidor, se preguntará la versión que se desea instalar: Windows o NetWare. En cualquier caso aparecerá la licencia de la versión de evaluación y se preguntará al usuario si desea instalar o desinstalar el software, o bien si desea crear discos de instalación. Desinstalar elimina los archivos de SQL Anywhere instalados anteriormente y la creación de discos de instalación copia la instalación del producto a discos para ordenadores sin unidad de CD-ROM. Si se elige instalar se pide el directorio destino y a continuación se permite la instalación de los componentes de desarrollo C/C++ y las librerías de enlace dinámico para aplicaciones, ambos opcionales. Posteriormente se procede a la copia de archivos y por último se realizan las modificaciones necesarias al AUTOEXEC.BAT y al CONFIG.SYS. Se puede optar por una modificación automática o por una modificación manual de estos ficheros. Para finalizar la instalación correctamente es necesario reiniciar el sistema. Las demás versiones se instalan de forma similar. Para más detalles sobre la instalación y requerimientos de los dis-





Click a Fund for More Info	Cur YTD	Annualized 1 Year	3 Years	5 Years	10 Years	Bull 10/90-9/95	Bear 5/90-10/90
Fidelity Blue Chip	$1.250	$1.201	$1.217	$1.253	N/A	$3.170	$864
Fidelity Contrafund	$1.316	$1.268	$1.193	$1.256	$1.200	$3.183	$882
Fidelity Dividend Growth	$1.289	$1.254	N/A	N/A	N/A	N/A	N/A
Fidelity Low-Priced Stock	$1.200	$1.176	$1.185	$1.253	N/A	$3.188	$866
Fidelity Magellan	$1.353	$1.296	$1.200	$1.226	$1.188	$2.645	$828
Fidelity OTC Portfolio	$1.365	$1.328	$1.168	$1.223	$1.169	$2.757	$836

tintos sistemas se recomienda consultar el archivo de texto \SQLANY\INSTALL.TXT. La documentación de Sybase SQL Anywhere se puede encontrar en forma de ficheros de ayuda dentro de los programas instalados.

VISUAL COMPONENTS

El directorio \VISUCOMP del CD-ROM incluye todos los archivos relacionados con Visual Components. Debido a que la instalación de todos ellos es muy parecida, solamente se explica en detalle el primer componente.

- Formula One:

Se trata de un componente que permite la manipulación de archivos de Excel. Proporciona soporte ODBC y es compatible con las versiones 4.0, 5.0 y 7.0 de Microsoft Excel.

Los archivos de Formula One se encuentran en el directorio \VISUCOMP\TRIALS\VCFORM1, dentro del cual aparecen los directorios OCX y SETUP. Para instalar Formula One se debe ejecutar el fichero SETUP.EXE que se encuentra en éste último directorio.

En primer lugar, este programa indica que se abrirá al instante de instalación, el cual nos guiará en dicho proceso. Para continuar se pulsa el botón Next y aparece una ventana desde la cual se puede cambiar el directorio de instalación (botón Browse) y se pueden elegir los elementos a instalar. Las opciones disponibles son: ficheros de control de 32 bits, ficheros de ayuda, ejemplos de Visual Basic 4, hojas de cálculo de ejemplo y ficheros de ejemplo de Visual C++ 4. Por defecto se encuentran seleccionados todos ellos. Para eliminar alguno se debe pulsar la marca situada a la izquierda del nombre del elemento, con lo que la marca desaparecerá. Si se desea volver a seleccionar algún elemento es necesario pulsar en el lugar donde se encontraba la marca. Para finalizar se pulsa el botón Next.

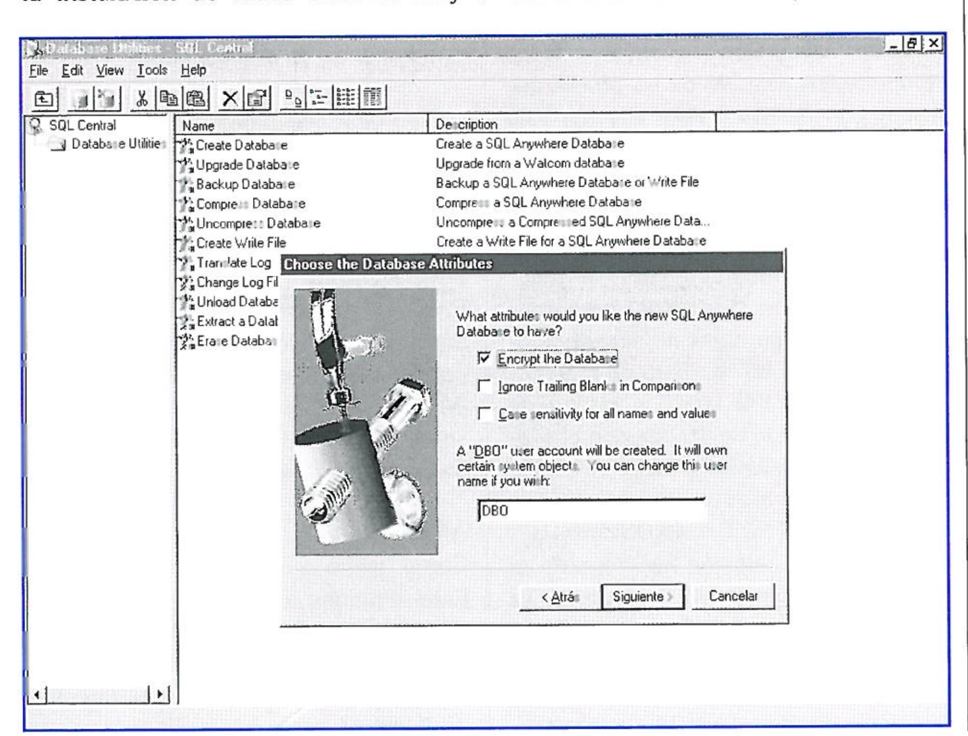
A continuación, el programa pregunta al usuario por el grupo de programas donde desea instalar Formula One. Pulsando Next de nuevo se procede a la instalación de los archivos. Por último se ofrece la posibilidad de leer el fichero README, con información importante sobre el producto. Se recomienda la lectura de dicho archivo.

- GeoPoint (directorio \VISUCOMP\TRIALS\VCGEOPT):

Este componente se utiliza para la presentación de datos mediante mapas. De forma análoga al componente anterior, el programa de instalación (SETUPEX.EXE) se encuentra dentro del directorio SETUP. En esta ocasión sólo se dispone de las siguientes opciones: ficheros de control de 32 bits, ficheros de ayuda y ejemplos de Visual Basic 4.

- First Impression (directorio \VISUCOMP\TRIALS\VCIMPRES):

First Impression es un componente destinado a la presentación de datos con diferentes tipos de gráficos. El programa de instalación es SETUP.EXE, y está situado en el interior del directorio SETUP. Las opciones de instalación dis-



sademo
File Command Window Help
Multi-user SQL Anywhere: sademo (2 users)
Userid Status Last Request Process
Configure
Disable connections
Continuous update
Logging to Message window
Multi-record blocking on fetches
Encryption of packets
Idle client shutdown time (minutes):
Quitting time (yy/mm/dd hh:mm):
Current Time: 96/10/09 01:13:00
Messages
Now acceptin
Start time: We
Starting check
Finished chec
Checkpoint Urgency
Checkpoint urgency 0
Recovery urgency 0
Temas de Ayuda: Sybase SQL Anywhere
Contenido | Índice | Buscar
Haga clic en un tema y después en Mostrar. También puede hacer clic en otra ficha, por ejemplo, en la ficha Índice.
Introduction to SQL Anywhere 5.0
How to Use This Book
New Features in SQL Anywhere 5.0
Introduction
What's in a name?
New features overview
New features in the Watcom-SQL language
New sample database
An Overview of SQL Anywhere
Tutorials
Using SQL Anywhere
Transact-SQL Compatibility
SQL Anywhere Programming Interfaces
Reference
Contact Information
Mostrar | Imprimir... | Cancelar

ponibles son las mismas del apartado anterior, más los ficheros de ejemplo de Visual C++ 4.

- VisualSpeller (directorio \VISUCOMP\TRIALS\VCSPELL):

Con VisualSpeller se puede añadir la función de corrección ortográfica a cualquier aplicación de Windows. Para instalarlo se ejecutará el programa SETUP.EXE, del directorio SETUP, y se seguirán los pasos descritos anteriormente.

- WebViewer (directorio \VISUCOMP\TRIALS\VCWEBV):

Como su propio nombre indica, WebViewer es un visor de páginas HTML implementado como control personalizado OLE. De nuevo, el programa de instalación recibe el nombre de SETUP.EXE, y está almacenado en el directorio SETUP.

- VisualWriter (directorio \VISUCOMP\TRIALS\VCWRITE):

Es un editor de textos WYSIWYG (*What You See Is What You Get*) que puede ser incorporado a cualquier herramienta de desarrollo con soporte de OCX. La instalación del mismo se realiza con el programa SETUP.EXE, del directorio SETUP.

La desinstalación de cualquiera de los componentes después de las pruebas se realiza con facilidad. Para ello tan sólo hay que utilizar el icono dispuesto a tal efecto en el grupo de programas correspondiente.

Además, se puede instalar la demostración de Formula One/NET, un plug-in para Netscape Navigator que permite la inclusión de hojas de cálculo activas en páginas HTML. Está en el directorio \VISUCOMP\F1NET y se puede instalar la versión de 16 bits con el programa \VISUCOMP\F1NET\F1PRO16\SETUP.EXE o bien la versión 32 bits \VISUCOMP\F1NET\F1PRO32\SETUP.EXE. En la instalación se pregunta al usuario por su nombre y empresa. El número de serie se debe dejar en blanco para instalar la demostración. Posteriormente se pide el nombre del directorio donde está instalado el navegador y se pide el nombre del directorio destino. Una vez finalizada la instalación se puede ver la hoja de cálculo de ejemplo LIVE.HTM, situada en el directorio SAMPLES del directorio destino indicado en la instalación.

También se incluye documentación en formato HTML en el CD-ROM.. La página raíz es TOUR.HTM y reside en el directorio \VISUCOMP.

BROWSERS

En el subdirectorio Browsers se encuentran diversos navegadores tanto para Linux, como para Windows 3.x y Windows 95. Para este último sistema se encuentra disponible E Internet Explorer 3.0 de Micrososft.

ARTÍCULOS

Como viene siendo habitual, el CD-ROM contiene los archivos correspondientes a los artículos de la revista. Se pueden encontrar en los distintos subdirectorios situados dentro del directorio \DISK27.

CORREO DEL LECTOR

En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relacionado con la programación.

ACCESO AL TECLADO

Ante todo felicitarles por su revista, la cual compro cada mes con agrado. Estoy realizando un programa escrito para Turbo C 2.0 que detecta la pulsación de cualquier tecla (incluidas teclas de función) y da su valor en decimal y ASCII. Mi problema es el siguiente: el programa funciona perfectamente cuando se pulsa la tecla, pero no cuando se suelta. ¿Sabrían ustedes alguna forma de detectar cuándo se suelta una tecla, dado que todas las funciones de Turbo C (kbhit (), etc) sólo detectan cuándo se pulsa?

Miguel Companyo Elias (Barcelona)

Efectivamente, existe una forma de detectar en qué momento el usuario suelta una tecla que previamente ha pulsado. Para comprender mejor el procedimiento, recordaremos a grandes rasgos el funcionamiento del teclado en los ordenadores PC.

Cada vez que se pulsa una tecla se guarda el código (ScanCode) de la tecla en el puerto 60h y se produce la interrupción hardware número 9. La rutina que atiende esta interrupción se encarga de tratar las combinaciones especiales de teclas como control+alt+del, introduce en el buffer de teclado los códigos ASCII de algunas de las teclas pulsadas y actualiza los flags de teclado situados en la zona de datos de la BIOS. Cuando se suelta una tecla sucede exactamente lo mismo, pero en el puerto 60h aparece el código de la tecla con el bit de mayor peso activo. Parece sencillo, pero hay un gran problema: las teclas expandidas y las teclas especiales. Las expandidas provocan que aparezca el valor 0E0h en primer lugar en el puerto 60h, y a continuación el código de la tecla. Existe una excepción, la tecla Pause, que produce un 0E1h en lugar de 0E0h. Las teclas especiales son equivalentes a la pulsación de dos teclas, es decir, cuando se pulsan y se sueltan producen 4 llamadas a la interrupción 9, en lugar de 2.

El siguiente programa de ejemplo muestra lo que ocurre con la interrupción de teclado. Debe compilarlo para 286 o superior.

```
#include <stdio.h>
#include <dos.h>

#define DIR_VIDEO 0xB8000000
#define PUERTO_TECLADO 0x60
#define INT_TECLADO 9

/*
Variable que contendrá el ScanCode de
la tecla
pulsada.
*/
unsigned char caracter;

/*
Puntero para escribir directamente en la
memoria
de video
*/
char far *punt_pant = (char far *)
DIR_VIDEO;

/*
Puntero a la rutina de servicio original
*/
void (interrupt far *vector_original)
(void);

/*
Nuestra rutina de interrupción
*/
void interrupt far mi_int_9 (void) {
  /* Ejecutar la rutina original */
  vector_original ();
   /* Guardar los registros para evitar
problemas */
  asm {
   pusha
  }
  caracter = inportb (PUERTO_TECLA-
DO);
  *punt_pant++ = caracter;
  *punt_pant++ = 14;
  /* Recuperar los registros de la pila */
  asm {
   popa
  }
}

int main () {
  char cadena [256];
  /* Colgar nuestra rutina de la interrup-
ción de teclado */
        vector_original = getvect
(INT_TECLADO);
  setvect (INT_TECLADO, mi_int_9);
  /* Leer una cadena para ver el efecto
de las pulsaciones */
  scanf ("%s", &cadena);
   /* Restablecer la rutina de atención
original */
  setvect (INT_TECLADO, vector_origi-
nal);
  return 0;
}
```


RESPUESTAS PREGUNTAS RESPUESTAS PREGUNTAS RESPUESTAS

A partir de este código le será muy fácil crear la rutina que necesita para detectar que una tecla ha sido soltada.

PROCESAMIENTO DE LOGS

P *Hola. Soy usuario de Linux y he comenzado a conectarme a Internet con este sistema operativo. Todo funciona bien, pero tengo un problema. Quiero capturar la dirección IP que me asigna mi proveedor en cada sesión para utilizarla en un script. Dicha dirección aparece en los logs del sistema nada más conectar. ¿Existe alguna forma de hacer esto?*

Julián Mateo Rincón
(Madrid)

R Lo más indicado es utilizar el AWK, un lenguaje pensado para la gestión de scripts. Asegúrese de tener dicho programa instalado en su sistema.

Nada más conectar puede ejecutar un script que contenga lo siguiente:

```
tail -n 50 /var/log/messages | awk -f
coger-ip > dir-ip.txt
```

Con esto tendrá la dirección IP grabada en el archivo dir-ip.txt y podrá incluirla donde quiera. El archivo coger-ip contendrá:

```
/local[  ]IP[ ]address/ {
  printf "%s", $9
}
```

Observe que en los primeros corchetes hay dos espacios y en los segundos tan sólo uno. En este ejemplo se ha supuesto que la dirección IP dinámica se vuelca en el archivo /var/log/messages. Consulte el archivo /etc/syslog.conf. Deberá contener una línea similar a la siguiente:

```
*.=info ;*.=notice        /usr/adm/messages
```

Una cosa más, si busca la ayuda del AWK con "man awk" no la encontrará. Utilice "man gawk" en su lugar.

RECONOCEDOR DE VOZ

P *Estoy preparando mi trabajo de fin de carrera que consiste en implementar un reconocedor de voz vía software. El programa lo estoy confeccionando bajo el entorno Windows usando el IDE de Borland C++ en su versión 4.5. Necesito información sobre cómo enviar mensajes entre programas, y quisiera que ustedes me recomendaran algún libro que la contenga.*

También necesito material bibliográfico que verse sobre la programación de tarjetas de sonido y su estructura interna (puertos, palabras de control, posibles interrupciones asociadas).

¿En qué números de Sólo Programadores han tratado estos temas? ¿Cómo puedo conseguirlos?

José Rubio Fernández
(Macael / Almería)

R Uno de los libros más prestigiosos sobre programación bajo Windows es, sin duda, "Programming Windows 3.1" de Charles Petzold, editado por Microsoft Press. En este libro podrá encontrar multitud de detalles sobre el funcionamiento interno de este entorno operativo.

En cuanto a libros sobre la programación de tarjetas de sonido, concretamente la Sound Blaster, le puede servir "El gran libro de la Sound Blaster", de A. Stolz. También puede pedir el kit de desarrollo Sound Blaster a la propia Creative Labs.

Sólo Programadores ha tratado en diversos números los temas que usted nos indica. En el nº 5 se habla de la programación de la Sound Blaster y se explica la técnica del doble buffer. El nº 9 contiene un artículo sobre reconocimiento de voz que puede serle útil. En el nº 14 se habla de la programación de la SB en general, y de la SB 16 en particular. Incluye un programa de ejemplo capaz de alterar el tono de voz en tiempo real. Un artículo muy técnico es "Programación del mixer de la SB", publicado en el nº 15. Otro artículo interesante es "Mejorando la calidad de sonido" del nº 16. En él se explica la interpolación mediante software para obtener una alta calidad de sonido en las tarjetas que no utilizan síntesis por tabla de ondas. Por último, puede facilitarle las cosas el artículo referente a los filtros digitales en tiempo real publicado en el nº 19. Puede pedir los números que le interesen a la editorial. La dirección y los números de teléfono y de fax puede encontrarlos en las primeras páginas de la revista.

ALTAVOZ DE PC

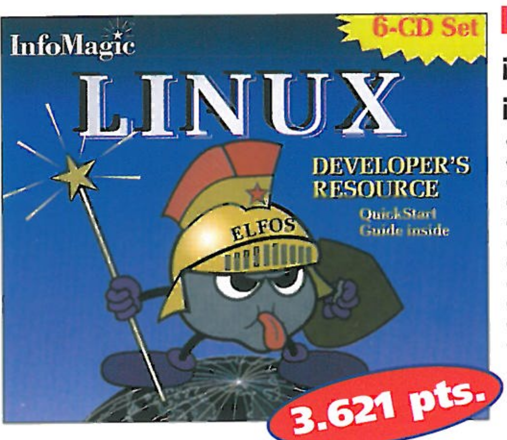
P *Hola, espero que me podáis contestar a estas preguntas: ¿Se pueden convertir ficheros de sonido MIDI a PC Speaker como con los PCM? ¿Cómo?*
Si se emplea el timer 1 para la reproducción de sonido digital a través del PC Speaker, ¿cómo se accede a la hora del sistema?

José Benito Castro Miguéns
(Cangas de Morrazo / Pontevedra)

R Realmente es posible reproducir un fichero MIDI por el altavoz del PC, pero requiere un esfuerzo considerable. En primer lugar sería necesario disponer de los distintos instrumentos digitalizados en archivos, de modo similar a los patches de tarjetas como la Gravis UltraSound. Por otro lado, habría que realizar un programa capaz de interpretar el formato MIDI, una tarea poco trivial. Y ahora viene la peor parte: mezclar correctamente los sonidos de los distintos instrumentos en memoria y realizar los cálculos necesarios para reproducir a través del altavoz con una calidad aceptable. Teniendo en cuenta los precios de las tarjetas de sonido en la actualidad, puede que no merezca la pena invertir mucho tiempo en una tarea como esta.

El timer 1 no se utiliza para la reproducción de sonido a través del altavoz. En un PC existen tres temporizadores programables. El timer 0 se utiliza para hacer saltar 18,2 veces por segundo la interrupción 8 (la del reloj). El timer 1 se utiliza para refresco de memoria y el timer 2 se emplea para generar sonido por el altavoz. Hecha esta aclaración, sólo queda decirle que dispone de varios métodos para acceder a la hora del sistema. Puede reprogramar la interrupción 8 y manejarla usted mismo, puede utilizar las funciones de la BIOS y también puede leerla directamente del área de datos de la BIOS.

CD-ROMs muy especiales



Linux Developer's Resource

¡Nueva edición Septiembre/96!
¡Ahora kernel 2.0! ¡6 CD-ROMs!

- Guía impresa de instalación rápida, 25 páginas.
- Red Hat 3.0.3 "Picasso" (kernel 1.2.13).
- Red Hat 3.0.3 para DEC Alpha.
- Slackware 3.1 (kernel 2.0).
- Debian GNU/Linux 1.1.4 (kernel 2.0).
- Fuentes del kernel hasta 2.0.12+.
- Xfree86 Version 3.1.2 (X-Windows).
- Archivos Linux de tsx-11.mit.edu y sunsite.unc.edu.
- Archivo GNU de prep.ai.mit.edu.
- Documentación on-line completa y HOWTO's.
- Demos comerciales, incluyendo: BRU, Cockpit, dBMan, Flagship, GP Modula-2, Smartware, Pathfinder, Scriptum...

Linux Slackware 96

¡Nueva edición Agosto/96!
¡Ahora kernel 2.0! ¡4 CD-ROMs!



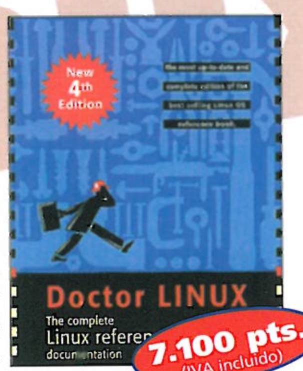
- Guía impresa de instalación rápida, 36 páginas.
- Distribución Slackware 3.1 (Walnut Creek).
- Incluye nuevo Kernel 2.0.
- Recopilación oficial por Patrick Volkerding.



Linux Red Hat Commercial v3.0.3

¡2 CD-ROMs + libro 190 pgs.!

- El Linux más solicitado en la actualidad.
- Versión oficial 3.0.3 de Red Hat Software, Inc.
- Incluye licencia para el servidor X de Metro.
- De fácil instalación y mantenimiento.
- Actualizaciones automáticas vía Internet y CD.





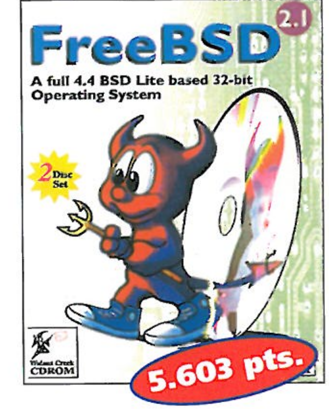
Doctor Linux 4ª edición

¡Libro de 1.600 páginas!

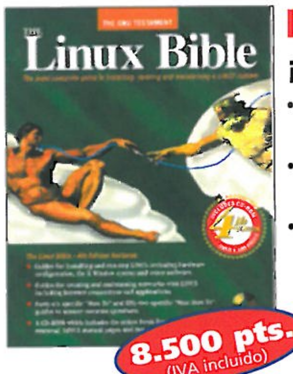
- El complemento ideal para el Linux Red Hat.
- Guías y libros de instalación, configuración, mantenimiento y administración.
- De fácil consulta, cuidada selección y organización.

FreeBSD 2.1

¡2 CD-ROMs + libro 300 pgs.!



- Sistema operativo UNIX completo y profesional.
- Basado en BSD 4.4 de Berkeley.



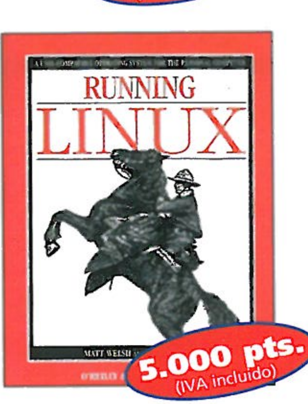
Linux Bible 4ª edición

¡CD-ROM + libro 1.800+ pgs.!

- La recopilación de documentación para Linux más completa existente.
- CD-ROM incluyendo el libro completo para búsquedas y consultas.
- Guías y libros de instalación, configuración, mantenimiento y administración.

Running Linux

¡El libro sobre Linux más popular y apreciado por los usuarios!
¡580 páginas!



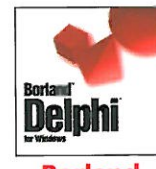
Sistemas Operativos/Programación: ***Disponemos del más amplio y especializado catálogo de CD-ROMs de utilidad para programadores y profesionales. Consúlte nuestro Web.***



Borland dBase v5.0 — **6.466 pts**



Borland Paradox v5.0 — **5.172 pts**



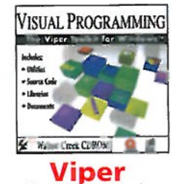
Borland Delphi — **6.466 pts**



Borland C++ v 4.0 — **4.224 pts**



Borland C++ v 4.5 — **6.466 pts**



Viper Herramientas de programación visual. — **3.879 pts**



Java Herramientas de programación en Java. — **2.586 pts**